TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

**Hannaliina Rakovitš**

# Programmeerimiskeele Python õpetamise töölehed III kooliastmele

**Bakalaureusetöö (9 EAP)**

Juhendajad: Tauno Palts, PhD
Riin Saadjärv, MA

Tartu 2023

## Programmeerimiskeele Python õpetamise töölehed III kooliastmele

**Lühikokkuvõte:** Üha enam on hakatud teadvustama õpilastele informaatika õpetamise tähtsust, kuna tänapäeva ühiskonnas on digitaalne kirjaoskus muutumas aina hädavajalikumaks. Selleks, et noored saaksid teha läbimõeldud valikuid oma edasises akadeemilises- ja igapäevaelus, on oluline, et nad saaksid lisaks muudele põhikoolis õpitavatele ainetele vähemalt sissejuhatuse informaatikasse. Selle bakalaureusetöö eesmärk on luua materjalid, mille abil saaks õpetada programmeerimist programmeerimiskeeles Python III kooliastme (7.-9. klass) õpilastele. Kursus koosneb seitsmeteistkümnest 90-minutilisest tunnist, mille jooksul antakse õpilastele baasteadmised informaatikast ja programmeerimisest. Kursuse vältel küsiti õpilastelt läbivalt tagasisidet koostatud materjalide jõukohasuse ja kursuse meeldimise kohta. Lisaks keskenduti küsimustikus õpilaste üldisele suhtumisele programmeerimisse. Selle tulemusena täiendati tehtud praktikumide materjale. Selgus, et töö raames koostatud kursus oli parajalt jõukohane III kooliastme õpilastele õppimaks programmeerimist. Samuti selgus, et kursuse tulemusena langes oluliselt õpilaste arusaam sellest, et programmeerimine on keeruline.

**Võtmesõnad:** Informaatika, põhiharidus, III kooliaste, riiklik õppekava, programmeerimise õpetamine

**CERCS:** P175 Informaatika, süsteemiteooria, S270 Pedagoogika ja didaktika

## Worksheets for Teaching Programming Language Python to Students in Third Stage of Study

**Abstract:** The significance of computer science education for students has been increasingly recognized in recent years due to the growing importance of digital literacy in the modern world. To enable young individuals to make informed decisions in both their academic and personal lives, it is crucial that they are introduced to computer science, along with other subjects taught in basic education. The objective of this bachelor's thesis is to formulate study materials that are intended for teaching programming to students in third stage of study (grades 7-9). The course is structured into seventeen 90-minute lessons, all aimed at providing students with basic knowledge of informatics and programming. The students were consistently requested to provide feedback on their enjoyment of the course and the course materials. The surveys conducted also assessed their general attitude towards programming. Based on the survey results, revisions were made to the initial practical materials. The study revealed that the programming course developed as a part of this work provided an appropriately challenging

learning experience for students in third stage of study. Additionally, it was evident that the course resulted in a considerable decrease in students' perception of programming being complicated.
**Keywords:** Informatics, primary education, third stage of study, national curriculum, teaching programming
**CERCS:** P175 Informatics, systems theory, S270 Pedagogy and didactics

## Sisukord

# 1. Sissejuhatus

Tänapäeva ühiskonnas on digitaalne kirjaoskus muutumas aina hädavajalikumaks, mille tõttu on viimastel aastatel hakatud üha enam teadvustama õpilastele informaatika õpetamise tähtsust. Programmeerimine ei ole väärtuslik oskus üksnes potentsiaalsel tulevasel töökohal, vaid see suurendab ka probleemide lahendamise võimet, loogilist mõtlemist ja loovust. Seetõttu on oluline programmeerimisõppe juurutamine hariduse varases staadiumis.

Võrreldes teiste erialadega Eesti ülikoolides, on informaatikaalastel erialadel enim väljalangemisi esimesel õppaastal [1]. Peamisteks põhjusteks on leitud vale eriala valik ja raskused programmeerimise õppimisel [2]. On täheldatud, et varem koolides läbitud informaatika- või arvutialased tunnid ei ole andnud üliõpilastele kõrgkoolides eelist või selliste tundide läbimise võimalus neil puudus [1]. Üldhariduskoolides informaatika õpetamise sisu on seni piirdunud vaid üldiste digipädevuste arendamisega.

Sarnaselt muudele põhikoolis õpetavatele ainetele annab informaatika õppimine õpilastele juurde pädevusi ja avardab silmaringi. Saab väita, et huvi informaatika vastu on seda suurem, mida rohkem on esinenud sellega kokkupuuteid. 2019. aastal tehtud uuringus [3] toodi välja, et 12. klassi õpilastel on suurem huvi informaatika vastu kui 9. klassi õpilastel just varasema kokkupuute tõttu. Seega on vaja rohkem informaatikaalaseid kursuseid, mida saaks üldhariduskoolides läbi viia, et suurendada kontakti enamikele põhikooliõpilastele seni tundmatu teadusega, nagu seda on informaatika.

Põhikooli lõpuklassides saadud haridus võib mängida kriitilist rolli õpilaste akadeemilises, isiklikus ja ka ametialases arengus. Selle bakalaureusetöö eesmärk on luua materjalid, mille abil saaks õpetada programmeerimist programmeerimiskeeles Python III kooliastme (7.-9. klass) õpilastele. III kooliastmes arenevad õpilaste loogiline mõtlemine ja analüüsivõime kiiresti ning nad on valmis ja paljudel juhtudel ka soovivad teha tutvust programmeerimisega.

Antud bakalaureusetöö aitab kaasa käimasolevale vestlusele programmeerimishariduse olulisusest hariduse varajases staadiumis ning näitab autori poolt välja töötatud materjalide tõhusust. Lõputöö tulemused on kasulikud haridustöötajatele ja poliitikakujundajatele programmeerimishariduse kavandamisel ja rakendamisel.

# 2. Programmeerimise õpetamine

Selles peatükis antakse ülevaade programmeerimise õppimise olulisusest ja programmeerimise õpetamise seisust Eesti haridusmaastikul. Tuuakse välja, millised väljakutsed pärsivad programmeerimise õppimist ja õpetamist ning arutatakse erinevaid viise programmeerimise õpetamiseks. Seejärel antakse ülevaade programmeerimiskeelest Python ja programmeerimiskeskkonnast Thonny. Peatüki lõpuks saavad lugejad paremini aru programmeerimise õpetamise eelistest ja väljakutsetest ning tõhusa programmeerimisõppe praktilistest strateegiatest.

## 2.1. Programmeerimise õppimise vajalikkus

Programmeerimise õppimise vajadus on aastatega muutunud järjest aktuaalsemaks. Tehnoloogia kujundab jätkuvalt meie igapäevaelu ja viimaste aastate kiire tehnoloogiline areng on toonud kaasa olulisi muudatusi paljudes eluvaldkondades, sealhulgas ka hariduses. Üks selline muudatus on kasvav nõudlus programmeerimisoskuste ja algoritmilise mõtlemise järele.

Algoritmiline mõtlemine (ingl *computational thinking*) koosneb probleemide lahendamise oskusest, süsteemide kavandamisest ja inimese käitumise mõistmisest, tuginedes informaatika põhiprintsiipidele [4]. Algoritmilise mõtlemise arendamine on tähtis iga indiviidi tasandil. Just selline mõtlemisstiil on ülioluline kujundamaks elus hädavajalikku kriitilist mõtlemist.

On märgatud, et algoritmilise mõtlemise arendamine toetab enesekindlust ja järjekindlust keeruliste probleemide lahendamisel [5]. Lisaks aitab see inimestel paremini meeskonnas töötada, tundes ära enda tugevused ja nõrkused. Kuigi enamik põhitõdesid programmeerimisel tulenevad matemaatikast, peavad paljud õpilased algoritmilist mõtlemist kõigest seni koolis õpitust erinevaks [6].

Sarnaselt matemaatikale, füüsikale ja keemiale, peaks iga inimene teataval määral oma elus programmeerimisega kokku puutuma. Tänapäeval on enamikel inimestel erinevaid nutiseadmeid, mis on selge märk sellest, et me ei sa informaatikat kui teadust ignoreerida. Kui muude reaal- ja loodusainete oskamist peetakse hädavajalikuks, siis maailmast täielikult aru saamiseks, peaks koolides õpetama ka programmeerimist [7].

Programmeerimine nõuab süsteemset ja loogilist mõtlemist, et luua probleeme lahendavaid programme. Seetõttu on see hea viis algoritmilise mõtlemise õpetamiseks. Vähemalt algelisel tasemel programmeerimisoskus on nõutud paljudes erinevates valdkondades, mitte ainult arvutiteaduste aladel. Kui üldhariduskooli eesmärk on anda õpilastele sellised teadmised, mis valmistaks nad tulevikuks ette, tuleks tänases maailmas märgata lisaks muudele reaal- ja loodusainetele ka programmeerimise õppimise vajalikkust.

## 2.2. Programmeerimine Eesti põhikoolides

Eesti Vabariigi põhikooli riiklikus õppekavas on ette nähtud informaatika õpetamine vaid valikaine raames [8]. Selle eesmärgiks on tagada põhikooli lõpetajale üldised digipädevused ehk peamine rõhk on turvalisel interneti kasutamisel, usaldusväärse informatsiooni hankimisel ning tekstifailide ja esitluste loomisel – programmeerimise õppimist määratud ei ole. 2023. aasta seisuga puudub põhikoolides ühtne ainekava informaatika õpetamiseks III kooliastmes – milliseid informaatikaalaseid aineid mingi kool pakub, on kooli enda otsustada. Selle tulemusena informaatika (sh programmeerimise) oskused õpilaste seas varieeruvad kooliti.

Võrreldes teiste erialadega Eesti ülikoolides, katkestatakse esimesel aastal õpinguid enim just informaatikaalastel erialadel [1]. Kuigi väljalangemise põhjuseid on erinevaid, on peamiseks põhjuseks leitud vale eriala valik – eraldi on välja toodud raskused programmeerimise õppimisel [2]. Selleks, et saada aru, mille poolest erinevad kõrgkoolidest väljalangenud ja õppimist jätkavad tudengid, intervjueerisid Kori ja Mardo [1] 16 üliõpilast – grupp oli jagatud pooleks katkestanute ja jätkanute vahel. Selle tulemusena saadi teada, et varasemalt koolides läbitud informaatika- ja arvutialased tunnid ei andnud üliõpilastele kõrgkoolides eelist või selliste tundide läbimise võimalus neil puudus. Üliõpilased nentisid, et informaatikat tuleks üldhariduskoolides kõigile õpilastele õpetada ühtlustamaks informaatikaalastele erialadele sisseastujate pädevusi.

Aastal 2019 avaldati uuring [3], mille raames vaadati, mis toimub Eesti üldhariduskoolides informaatika õpetamise tasandil. Selles uuringus jäeti informaatika definitsioonist välja üldised digipädevused, mis on põhikooli riiklikus õppekavas informaatika valikaines [8] kesksel kohal, ja keskenduti peamiselt informaatikasse kui teadusesse. Uuringus osalenud õpilased ja õpetajad tõid välja, et seni on arvutiõpetuse ja informaatika tundide sisu piirdunud peamiselt siiski digipädevuste arendamisega.

Uuringus [3] osalenud õpilastest 37% soovisid end tulevikus informaatikaga siduda – ei ole selge, kui suur osa neist kuulus III kooliastmesse. Leiti, et soov informaatikaga tegeleda sõltus suuresti varasemast huvist. Ilmnes, et 9. klassi õpilaste huvi informaatika vastu oli väiksem kui 12. klassi õpilastel just varasema kokkupuute tõttu. Uuringus osalenud õpetajad tõid välja, et põhikooli õpilasel puudub piisav ettevalmistus informaatika vallas. Lisaks sellele mainiti vajadust rohkemate informaatikaalaste kursuste järele, mida saaks vajadusel koolis läbi viia.

Tartu Ülikooli teadlased andsid mainitud uuringus [3] välja soovitusi, kuidas parandada Eesti põhikooli õpilaste informaatikaalaseid oskuseid. Üks soovitus oli panna rohkem rõhku põhikooli õpilaste informaatikaalase huvi tõstmisele. See võiks hõlmata näiteks praktilisi programmeerimisülesandeid ja -projekte, mis võimaldaksid õpilastel paremini mõista, kuidas informaatikat igapäevaelus kasutatakse. Samuti soovitatakse luua rohkem õppematerjale, mis oleksid põhikooli õpilastele kättesaadavad ja aitaksid neil paremini mõista informaatika põhimõtteid. Oluline on ka põhikooli lõpetajate informaatikaalaste oskuste ühtlustamine, mis võiks olla võimalik, kui osa õppest viiakse läbi juba põhikooli tasemel. Kõik need soovitused võiksid aidata kaasa informaatikaalase hariduse kvaliteedi parandamisele ja tagada, et rohkem noori Eestis teeks teadlikke tulevikuplaane.

## 2.3. Programmeerimise õppimise ja õpetamise keerukus

Programmeerimine on tänapäeva digimaailmas võtmeoskus. Üha enam nõutakse lisaks arvutiteaduste aladele ka muudes valdkondades programmeerimise oskust. Ei ole võimalik ühtselt hinnata, kui lihtne või keeruline on programmeerimist õppida ja õpetada. Kuigi programmeerimise õpetamisel noortele on palju positiivseid külgi, ei tule see väljakutseteta.

Jääb mulje, et eksisteerib ebakõla selle vahel, mida õpilased ja õpetajad programmeerimise õppimise juures keeruliseks peavad. Programmeerimisega alustades peavad õpilased tihedamateks probleemideks süntaksvigu ja veateadete mõistmist, kuid õpetajad toovad murekohtadena välja raskused loogika ja aritmeetika tehetega [9]. Samas on raskustena välja toodud ka funktsionaalsuse jagamine osadeks ja vigade leidmine enda kirjutatud koodis [10]. Sellised keerulised väljakutsed võivad tekitada tingimused, mis ei motiveeri algajaid programmeerijaid õpingutega jätkama.

Siiski ainult motiveeritud õpilastest ei piisa. Sarnaselt igale muule õpitavale ainele, on hea õpetaja programmeerimises kriitilise tähtsusega. Üheks suurimaks väljakutseks ka ülikoolide programmeerimise algkursustel on pädevate ja entusiastlike juhendajate leidmine. Olukorras, kus õpetajaks on inimene, kes ei tunne end programmeerimisega mugavalt, on võimalus, et õpilastele jääb informaatikast kui teadusest läbivalt negatiivne mulje [11].

Õpetajad seisavad silmitsi mitme väljakutsega. Nad peavad olema võimelised toetama erinevaid õpivajadusi ja lisaks leidma veel aega, et anda õpilastele adekvaatset tagasisidet. Selleks, et õpetajad saaksid oma tööd paremini teha, on oluline neid toetada. Üks variant selleks on teha neile kättesaadavaks õppematerjalid, mis on abiks erinevate teemade õpetamisel. Aastal 2019 avaldatud uuringus [3] nentisid õpetajad, et eksisteerib vajadus rohkemate informaatikaalaste kursuste järele, mida saaks vajadusel koolis läbi viia.

## 2.4. Programmeerimise õpetamise viisid

Kõige traditsioonilisem viis programmeerimist õpetada on osa õppetööst läbi viia loengu vormis. Seda viisi kasutatakse tänase päevani paljudel programmeerimise algkursustel. Sarnaselt õpetajale, on ka õpilastel suur roll klassiruumis. Selle meetodi suurim eelis on õpilaste füüsiline osalemine õppetöös – hariduse omandamisel on oluline selle tegemine koos teistega. Selline lähenemine annab võimaluse vajadusel küsida küsimusi ja suhelda õpetaja või klassikaaslastega. Kuigi traditsioonilised meetodid nagu loengud ja õpikud võivad olla efektiivsed, ei pruugi need kõigile sobida.

Probleemipõhises õpetamises (ingl *probleem-based learning*) on kesksel kohal õpilastele väljakutsetena probleemide esitamine ja nende lahendamine kujundabki õppimisprotsessi [12]. Tegu on praktilise meetodiga ja eriline rõhk on teadmiste omandamisel tegemise kaudu. Õpilastele esitatavad probleemid võivad olla lihtsad ülesanded stiilis „Koosta programm, mis...“. Seda meetodit on edukalt rakendatud sissejuhatavatel programmeerimiskursustel. Lisaks probleemide lahendamise oskuse õpetamisele, annab see efektiivselt edasi programmeerimise põhiprintsiipe [13].

Projektipõhine õpe (ingl *project-based learning*) on õppemeetod, mida iseloomustab autonoomia, eesmärkide seadmine, koostöö ja suhtlemine [14]. Selle tulemusena areneb õpilaste kriitilise mõtlemise, ajaplaneerimise ja koostöövõime oskused. Aastal 2021 avaldatud

uuringus [15] täheldati, et võrreldes traditsiooniliste õppemeetoditega, saavutati projektipõhise õppe tulemusena märgatavalt paremad tulemused. Lisaks on välja toodud, et selline õppevorm suurendab õpilaste entusiasmi õppimisprotsessis [16].

Paarisprogrammeerimine (ingl *pair programming*) kujutab endas kaht kõrvuti olevat õpilast lahendamas ühist ülesannet jagatud arvuti taga. Selles kujuneb välja kaks rolli: juht, kes kontrollib arvutit, ja juhendaja, kes ekraani jälgides annab ettepanekuid ja soovitusi. Aastal 2011 avaldatud uuringus [17] võrreldi paaris ja individuaalselt töötanud õpilaste hindeid. Õpilased, kes töötasid paaris, said märgatavalt paremaid tulemusi kui need, kes kursuse töö individuaalselt läbisid. Õpilased on andnud selle õppemeetodi kohta palju positiivset tagasisidet. Nende sõnul aitab see säilitada häid õpiharjumusi ning pakub õppeprotsessis väärtuslikku tuge ja julgust. 2014. aastal avaldatud uuring [18] leidis, et õpilaste endi arust lahendasid nad ülesande kiiremini võrreldes sellega, kui oleksid seda teinud iseseisvalt.

## 2.5. Programmeerimiskeel Python ja programmeerimiskeskkond Thonny

Python on interpreteeritav programmeerimiskeel, mis avaldati aastal 1991. Aastatega on selle kasutatavus aina kasvanud tänu suurenenud nõudlusele andmetöötluse ja masinõppe järele. Aastal 2022 peeti Pythonit teiseks kõige populaarsemaks programmeerimiskeeleks TIOBE indeksi alusel [19].

Programmeerimise õppimisega alustades on oluline keskenduda programmeerimise kontseptsioonidele mitte keele spetsiifikale [20]. Seetõttu on Python sobivaim keel algajatele. Tänu koodi lühidusele, sisseehitatud kõrgetasemelistele andmetüüpidele, suurele teekide kogusele ja lihtsale süntaksile on see enim kasutatud keel ülikoolide programmeerimise algkursustel [21]. Pythoni lihtne süntaks on arusaadav igale inglise keelt kõnelevale ja põhikooli tasemel matemaatikat oskavale inimesele.

Thonny on Tartu Ülikoolis välja töötatud tasuta allalaaditav Pythoni arenduskeskkond. Annamaa [22] väidab, et see on algajasõbralik arenduskeskkond, mille abil programmeerimist õppida ja õpetada. Tema sõnul on Thonnyl palju eeliseid – neist silmapaistvaimad on võimalused koodi silumiseks. Silumine (ingl *debugging*) on protsess, mille abil tuvastatakse ja parandatakse koodis vigu [23]. See on algajale programmeerijale üks olulisemaid tegevusi mõistmaks, kuidas kirjutatud kood töötab.

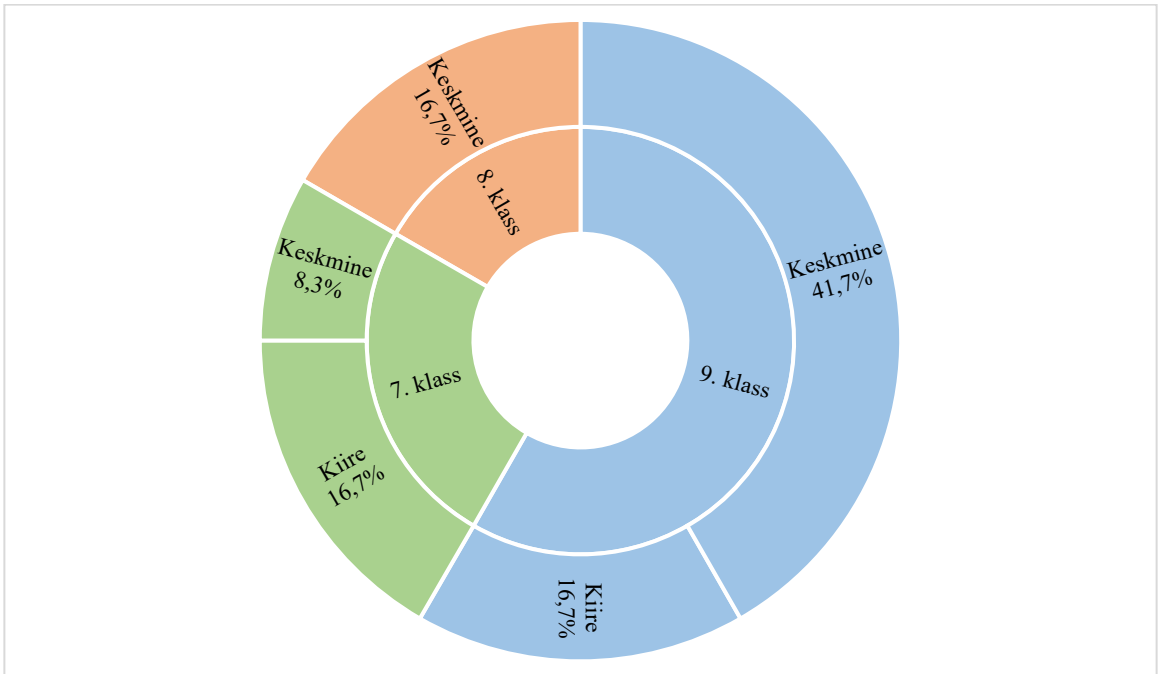
Võrreldes teiste programmeerimiskeskkondadega, näiteks PyCharm ja Atom, on Thonny kasutamine lihtne ja intuitiivne – seal puuduvad liigsed funktsioonid, mida algaja programmeerija ei vaja. See võimaldab õppijal keskenduda vaid algoritmilise mõtlemise oskuse ja programmeerimise põhitõdede omandamisele.

## 3. Metoodika

Selles peatükis antakse ülevaade III kooliastme õpilastele mõeldud programmeerimiskursuse koostamise ja tagasiside kogumise protsessist. Samuti tuuakse välja, mis valimi peal kursus läbi viidi. Peatükis kirjeldatakse üksikasjalikult, kuidas küsitlusi korraldati ja milliste meetoditega saadud tulemusi analüüsiti.

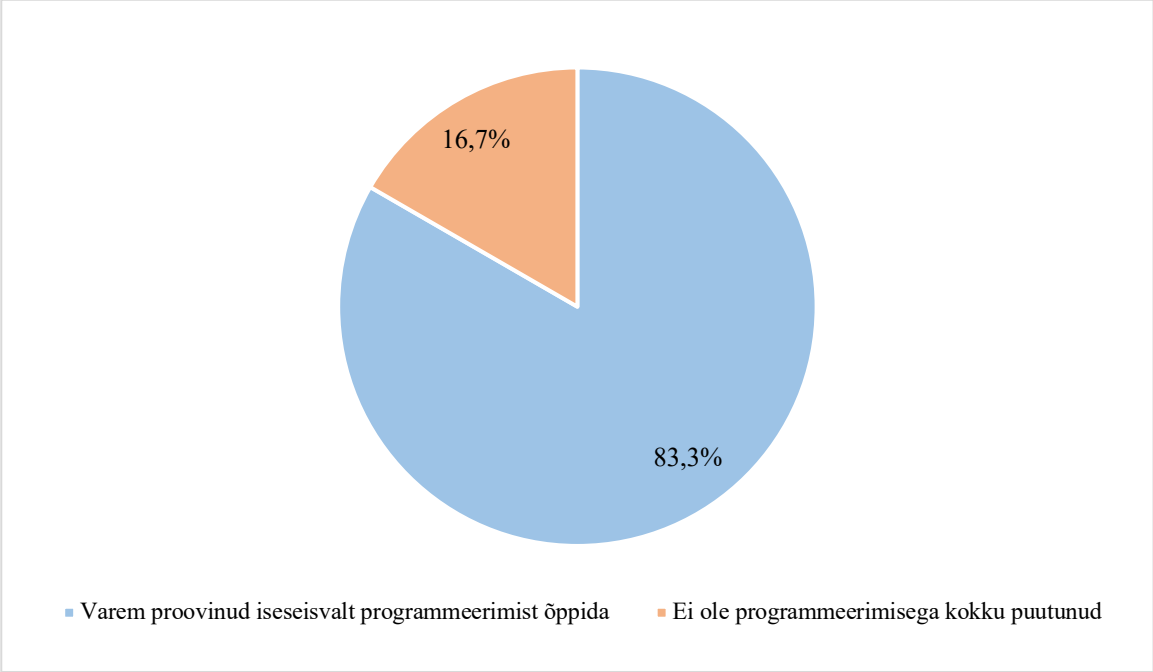
### 3.1. Valim

Bakalaureusetöö raames koostatud kursusel osales vabatahtlikkuse alusel 12 õpilast, kellest seitse (58,3%) käis 9. klassis, kaks (16,7%) käis 8. klassis ja kolm (25,0%) käis 7. klassis. Osalenud õpilastest kolm (25,0%) olid tüdrukud ja ülejäänud (75,0%) poisid. Praktikumid viidi läbi ühes Tartu koolis, kus on omane jaotada õpilased reaal- ja loodusainetes temporühmadesse. Nende temporühmade nimetused on „aeglane“, „keskmine“ ja „kiire“ ning õpilase kuuluvuse ühesse nendest gruppidest määrab see, kui kiiresti omandab ta õppekavas ettenähtud teemad. Enamik õpilasi (66,7%) kuulus temporühma „keskmine“ ja ülejäänud (33,3%) kuulusid temporühma „kiire“. Joonisel 1 on välja toodud õpilaste jaotumine nii klassidesse kui temporühmadesse.




Joonis 1. Õpilaste jaotumine klassidesse ja temporühmadesse.

Antud valimist enamus õpilasi (83,3%) oli enne kursusega alustamist proovinud iseseisvalt õppida programmeerimist, kaks õpilast (16,7%) ei olnud varem kordagi programmeerimisega kokku puutunud. Üheksal õpilasel (75,0%) oli varasem kogemus visuaalse programmeerimiskeelega Scratch ja üks õpilane (8,3%) oli eelnevalt proovinud õppida ka Pythonit. Joonisel 2 on välja toodud õpilaste varasema programmeerimisega kokkupuutumise osamäärad.




Joonis 2. Õpilaste varasem kokkupuude programmeerimisega.

## 3.2. Kursuse ja õppematerjalide koostamise protsess

Kursuse koostamise protsess algas aasta 2022 augusti alguses. Esmalt viidi end kurssi sellega, millised erinevad õpetamise metoodikad eksisteerivad ja mis on need teemad, mida algaja programmeerija oskama peaks. Uurimise tulemusena otsustati koostatava kursuse raames õpetada alltoodud teemasid.

- Sissejuhatus programmeerimisse – programmeerimiskeeled, programmeerimisega seotud tööpõhimõtted, programm, protsess ja roll.
- Algoritm – algoritmi mõiste, algoritmi koostamine, algoritmi realiseerimine, etteantud tegevusjuhisest arusaamine, tegevusjuhise ise koostamine, tegevusjuhise rakendamine, andmete otstarbekalt muutmine ja lihtsamate tüüpalgoritmide kasutamine.
- Andmed – objektide omadused ja kasutamine.

- Muutujad – muutuja kasutamine ja sellele väärtuse andmine.
- Sisendid ja väljundid – sisendi andmine klaviatuurilt.
- Tegevused ja avaldised – lihtsamad teksti-, loogika- ja arvavaldised.
- Valikud – tingimuslaused *if*, *elif* ja *else*.
- Kordused – lõpmatu kordus, kordamine teatud arv kordi, kordamine etteantud tingimustel ja kordus korduse sees.

Seejärel tehti kindlaks, kuidas kursuse raames õppematerjale koostada. Otsuses arvestati olukorraga, et aasta 2022 augusti seisuga puuduvad avalikult kättesaadavad kursused, mis oleks suunatud III kooliastmele ja sarnaneks selle bakalaureusetöö raames koostatavale kursusele. Selle tulemusena otsustati praktikumide materjalid koostada jooksvalt kursuse vältel – pärast iga toimunud praktikumi koostati järgmise tunni materjalid. See andis võimaluse täpselt hinnata, kuidas III kooliastme õpilased eelnevalt väljatoodud teemasid omandavad – millises tempos neid läbima peaks, et õppeprotsess oleks tõhus.

## 3.3. Tagasiside kogumine ja andmeanalüüs

Enne kursuse algust koguti õpilastelt teavet nende suhtumiste ja harjumuste kohta (Lisa I). Andmeid koguti õpilastelt alltoodud väidete kohta. Hindamine toimus Likerti skaalal, kus „1“ väljendas täielikku mittenõustumist ja „5“ väljendas täielikku nõustumist. Seejuures lisati juurde ka variant „0“ juhuks, kui õpilane ei saa väitele enda kohta vastata.

- „Olen huvitatud programmeerimise õppimisest.“,
- „Olen väga elevil programmeerimise õppimisest.“,
- „Arvan, et programmeerimine on raske.“.
- „Kasutan igapäevaselt oma telefoni, tahvelarvutit või arvutit.“,
- „Mulle meeldib matemaatika.“,
- „Mulle meeldib lahendada mõistatusi ja ristsõnu.“,
- „Mulle meeldib mängida malet.“.

Iga kursuse jooksul läbitud praktikumi lõpus koguti õpilastelt andmeid kursuse materjali, tunnikorralduse ja programmeerimisega seotud arvamuste kohta. Selles protsessis kasutati ühtlast küsimustikuvormi (Lisa II), mis sisaldas alltoodud väiteid. Need küsimustikud olid mõeldud mõistmaks materjali raskusastet III kooliastme õpilaste jaoks. Olulisel kohal oli veel

õpilaste üldine õpikogemus koostatud kursuse vältel. Hindamine toimus Likerti skaalal, kus „1“ väljendas täielikku mittenõustumist ja „5“ väljendas täielikku nõustumist.

- „Mulle meeldis tänane praktikum.“.
- „Ülesanded olid minu jaoks jõukohased.“.
- „Jõudsin kõik ülesanded valmis.“.
- „Arvan, et programmeerimine on raske.“.
- „Ootan juba järgmist programmeerimise praktikumi.“.

Kursuse lõpus viidi õpilastega läbi avatud struktureeritud küsimustik. Saadud tagasiside alusel püüti mõista, kuidas õpilased hindavad kursust ja enda tulemuslikkust peale selle läbimist. See võimaldas lisaks analüüsida kursuse kitsaskohti, et üha enam kohandada kursuse sisu III kooliastme õpilastele. Informatsiooni koguti alltoodud küsimuse abil.

- „Kas arvad, et sul läks sellel kursusel hästi? Mida oleksid saanud teha, et kursusel paremini läheks?“,
- „Mis sulle kõige rohkem meeldis?“,
- „Mis sul kõige vähem välja tuli?“,
- „Mida sa sellel kursusel kõige enam nautisid?“,
- „Kas plaanid tulevikus veel programmeerimisega kokku puutuda?“.

Peale teema täielikku läbimist hindasid õpilased Likerti skaalal, kui keeruliseks nad läbitud teemat peavad. Hindamisel „1“ väljendas teema pidamist väga lihtsaks ja „5“ väljendas teema pidamist väga keeruliseks. Seda tehti iga kursusel läbitud teema kohta, seejuures omavahel seoti teemad andmetüübid, muutujad, tehted ja kasutaja sisend. Nende teemade läbimine toimus mitme praktikumi vältel paralleelselt ja õpilastel ei olnud võimalik neid teineteisest eristada.

Töös analüüsiti õpilaste tagasisideküsitlustest kogutud andmeid, kasutades kirjeldavat statistikat. Selle abil üldistati ja kirjeldati saadud andmeid. Peamiselt hõlmas iga uuringuküsimuse analüüs keskmiste arvutamist koos standardhälbega. Seda kasutati selleks, et tuvastada õpilaste suhtumist programmeerimise õppimise ja kursuse materjali osas. Andmeanalüüsi tulemusi kasutatakse antud töö raames koostatud kursuse tulemuslikkuse hindamiseks ja praktikumide materjalide täiustamiseks.

# 4. Programmeerimise kursus III kooliastme õpilastele

Selles peatükis antakse ülevaade bakalaureusetöö raames koostatud programmeerimise kursusest. Lähemalt on kirjutatud kursuse ainekavast ja praktikumide materjalide olemusest. Peale selle peatüki lugemist on lugejal selge arusaam, milline näeb välja bakalaureusetöö raames koostatud programmeerimise kursus III kooliastme õpilastele.

## 4.1. Programmeerimise kursuse ainekava

Programmeerimise kursus on ettenähtud III kooliastmele ja selle ainekava (Lisa III) jaotab omandatavad teadmised ära seitsmeteistkümne 90-minutilise tunni vahel. Kursuse struktuur on üles ehitatud selliselt, et õppetöö toimuks keskkonnas, kus on koheselt võimalik abi saada – kogu õppetöö toimub praktikumides, kodutöid antud kursuse raames ette nähtud ei ole. Kursusel kasutatakse programmeerimiskeelt Python ja programmeerimiskeskkonda Thonny.

Kursuse peamine eesmärk on anda õpilastele esimene kogemus programmeerimisega. Kursust lõpetav õpilane

- mõistab ja kasutab teadlikult erinevaid programmeerimisega seotud mõisteid,
- on suuteline programmi käitumist enne käivitamist ennustada ja programmi töö tulemust analüüsima,
- oskab programmides teha otstarbekaid muudatusi ja/või täiendusi,
- oskab koostada programmi sõnalise kirjelduse alusel,
- oskab koostada lihtsamaid avaldisi ja algoritme,
- oskab parandada vigu olukorras, kus programm ei tee talle soovitud tööd.

Enne kursuse algust pandi paika algne töökava, kuid lõplik ainekava (Lisa III) valmis selle inspiratsioonil, analüüsides jooksvalt õpilaste võimekust ettenähtud teemad omandada. Selle tulemusena on kursuse raames kohustuslik omandada peatükis „3.2. Kursuse ja õppematerjalide koostamise protsess“ välja toodud teemad.

Ainekavas (Lisa III) on kõikide praktikumide kohta välja toodud tunnis omandatavad põhimõisted, tunnis toimuv teoreetiline ja praktiline tegevus, soovitavad õpitulemused, hindamine, lõiming muude ainetega ja praktikumi läbiviimisel kasutatav materjal. Kõik praktikumide materjalid on koostatud antud töö raames, jälgides spetsiifiliselt, kuidas III kooliastme õpilased programmeerimisega seotud teadmisi omandavad.

## 4.2. Programmeerimise kursuse praktikumid

Praktikumid viis läbi antud bakalaureusetöö autor, kes arvestas kursuse vältel materjalide väljatöötamisel õpilaste õppimistempoga. Kursuse vältel jälgiti, et materjalid oleksid õpilastele arusaadavad ja kaasahaaravad, andes samal ajal väljakutseid pakkuvat õppimiskogemust. Vastavalt algsele plaanile tehti materjalid igaks praktikumiks valmis vahetult peale eelmise praktikumi toimumist. Selliselt oli võimalik analüüsida, kas läbitud teemad on õpilastele arusaadavad.

### 4.2.1. Praktikum 1

Esimese praktikumi materjalide (Lisa IV) koostamisel oli kesksel kohal õpilastele programmeerimisse ja programmeerimise tööpõhimõtetesse sissejuhatuse andmine. Autor esitas informaatika vallas levinud väljendid, kasutades III kooliastme õpilastele tuttavaid teadmisi ja kontseptsioone. Terminid nagu algoritm, plokkskeem ja programm, on programmeerimisest rääkides pidevas kasutuses, ka loodud kursuse edasistes praktikumides. Iga termin illustreeriti võimalusel visuaalse näitega ja selle kinnistamiseks anti õpilastele lahendamiseks ülesanne, kus seda teadmist kasutada tuli.

Algoritmi ja plokkskeemi teadmise kinnistamiseks koostati ülesanne 1. Selle raames on oluline õpilaste teadvustamine, et koostavad algoritmi ja väljendavad seda visuaalselt plokkskeemi abil. Programmi olemusest arusaamiseks on ettenähtud õpilastele tutvustada programmeerimiskeelt Python ja programmeerimiskeskkonda Thonny. Seejärel tuleb koostada koos õpetajaga ülesandes 2 koos esimene programm Turtle mooduli abil. Turtle moodul võeti kasutusse selleks, et õpilastel oleks koheselt visuaalne näide programmi töötamisest – iga rida on programmis uus käsk, mida arvuti täidab.

Kõik esimeses praktikumis koostatavad programmid tehakse Turtle mooduli abil. III kooliastme õpilased on õppinud piisavalt inglise keelt, et seal kasutatavad väljendid nagu *forward*, *backward*, *right*, *left*, *up*, *down*, *speed*, *color*, *begin_fill* ja *end_fill* on neile arusaadavad. Sellegipoolest ühtlustamaks materjalist arusaamist, tehti töölehele ülevaade Turtle'i käskudest nende tegevuslike eestikeelsete vastetega.

Ülesandes 3 peavad õpilased joonistama Turtle mooduli abil ette antud kujundid. 7. klassis omandatakse teadmised korrapärase hulknurga sisenurkade summast ja ühe sisenurga

suurusest. Antud ülesande loomisel võeti aluseks, et III kooliastme õpilane riikliku õppekava alusel peab olema võimeline eelnimetatud suurusi arvutama. Siiski taseme ühtlustamiseks pandi töölehele nii vihjed kui ka visuaalne näide tehete ja selgitustega.

Praktikumi viimaseks tegevuseks on ette nähtud lihtne loominguline ülesanne. Ülesandes 4 peavad õpilased joonistama vabalt valitud pildi. Selle eesmärk on lõpetada praktikum lihtsa ülesandega, mis illustreerib, et väljaspool programmeerimiskeele süntaksit puuduvad programmeerimises ranged piirid ning sellega saab teha ja väljendada kõike, mida nad soovivad.

### 4.2.2. Praktikum 2

Teise praktikumi materjalide (Lisa V) koostamisel oli kesksel kohal anda õpilastele teadmised, millele reaalne rakendus tuleb alles kolmandas praktikumis. Andmetüübid, muutujad ja tihti ka tehted esinevad igas koostatavas programmis ning nendest ja nende praktilistest rakendustest aru saamine on kriitilise tähtsusega programmeerimise õppimise jätkamisel.

Töölehtedel selgitatakse õpilastele Pythonis esinevad tihti kasutatavad algelised andmetüübid varem matemaatikas ja eesti keeles õpitud terminite abil, iga näide on illustreeritud mitme näitega. Visuaalselt on välja toodud viis, kuidas kontrollida, mis andmetüübiga on tegu. Selle jaoks tehakse ka esimene tutvus *print* käsuga, millele on kogu kursuse vältel igas praktikumis kasutus. Lisaks on visuaalse näite abil selgitatud õpilastele, kuidas teha teisendusi andmetüüpide vahel. Viimase kinnistamiseks ja lisanuputamiseks on praktikumi lehtedele pandud ülesanne 1. Antud ülesandes on kasutatud teadmist ümardamisest, millel on seos varem arvutiõpetuse tundides õpitud Exceli *round()* funktsiooniga.

Põnevaks nuputamiseks ja andmetüüpide omadustele keskendumiseks pandi praktikumi lehtedele ülesanne 2. Antud ülesandes peavad õpilased aru saama, millise andmetüübiga on tegu. Ülesandes antud andmetüüpe on esmapilgul lihtne pakkuda, kuid sisusse süvenemata võib õpilane eksida. See ülesanne pandi selleks, et õpilased harjuksid vigade tegemise abil süvenema kirjutatud koodi.

Visuaalse koodinäite abil on praktikumi lehtedel ära selgitatud muutuja omadus ja kuidas seda väärtustada, eraldi on välja toodud reeglid muutujatele nimede valimisel. Muutuja

väärtustamine on õpilastele tuttav 7. klassis õpitud funktsioonide teemast. Muutujatele nime andmise kinnistamiseks on ülesanne 3, kus õpilased peavad hindama, kas antud nimi sobiks programmeerimiskeeles Python muutuja nimeks.

Aritmeetikatehted ja tehete järjekord on õpilastele tuttavad varasematest õpingutest, kuid taseme ühtlustamiseks on need lehtedel eraldi välja toodud. Põnevaks nuputamiseks on õpilastele antud ülesanded 4, 5 ja 6. Ülesannetes 4 ja 6 peavad õpilased tegema antud tehted ja saama vastused pidades meeles praktikumi alguses õpitud andmetüüpe.

Teise praktikumi lõpetab Turtle mooduli abil joonistamine. Sarnaselt esimesele praktikumile saab õpilane vabalt valida, mida joonistada soovib, kuid sel korral tuleb arvestada, et käskudes kasutatavad suurused tuleb eelnevalt salvestada muutujatesse. Selliselt harjuvad õpilased juba õpitu baasil ära muutujate väärtustamise ja kasutamisega.

### 4.2.3. Praktikum 3

Kolmandas praktikumis (Lisa VI) rakendatakse eelmises praktikumis omandatud teadmisi lühikeste programmide koostamisel. Selle jaoks selgitatakse praktikumi materjali alguses õpilastele lisaks, kuidas saada kasutajat oma programmiga suhtlema. Kasutajalt sisendi küsimine on programmeerimises levinud kontseptsioon ehk selle juurutamine õpingute varajases staadiumis on kriitilise tähtsusega.

Õpilased saavad omandatud teadmisi rakendada antud ülesannetes, kus oodatavad väljundid on illustreeritud piltidega. Ülesannetes on oluline eelmisest praktikumist meelde tuletada, mis on muutuja ja kuidas sellele väärtust anda. Lisaks tuleb meenutada, kuidas ekraanile tulemusi väljastatakse ja mida selleks silmas peab pidama. Eelmises praktikumis õpitud tehted ja ümardamise oskus tuleb nüüd reaalselt programmi koostamise näol kasutusse panna. Üks ülesanne on lõimitud matemaatikas õpitud kujundi ümbermõõdu ja pindala valemite teadmisega.

Selleks, et õppimine oleks põnev kõigile, tuleb kiirematele anda keerulisemaid ülesandeid. Esimest korda kursuse vältel on praktikumide materjalides lisaülesanded. Nende lahendamiseks on antud õpilastele vihjeid ja nad on julgustatud iseseisvalt otsima internetist

informatsiooni. Kui õpilasel jääb praktikumi põhiülesannetest aega üle, näitab see seda, et ta on valmis kursuse vältel olema oluliselt iseseisvam võrreldes teistega.

Lisaülesanded on 9. klassi matemaatikas õpitava baasil. Selleks tuleb tutvuda Pythagorase teoreemi ja ruumiliste kujunditega. Informatsioon nende kohta on kättesaadav internetist ja ka praktikumi juhendajalt. On täiesti reaalne, et mõni õpilane ei lahendagi neid ülesandeid ära, ja seepärast ei ole need põhiülesannete hulgas.

### 4.2.4. Praktikum 4

Neljas praktikum (Lisa VII) on üles ehitatud selliselt, et õpilased saaksid aru, mis on tingimuslause. Oluline on mõista tingimuslauset igapäeva elu baasil – otsuste tegemine on osa iga inimese päevast. Selle algoritmiliselt läbi mõtlemine on üks programmeerimise põhioskustest.

Praktikumi materjalis on selgitatud, mis on tingimuslause. Näidete ja esimeses praktikumis õpitud plokkskeemi abil on selgitatud tingimuse olemus. Õpilased saavad ülesandes 1 tuletada meelde plokkskeemi, kuid sel korral endale teadvustades, et küsimus enne otsustamist on tingimuslause. See ülesanne teeb õpilasele selgeks, et tingimuslause pole nende jaoks tegelikult uus kontseptsioon.

Illustreerivate näidete ja detailsete selgitustega on materjalides ära näidatud, kuidas programmeerimiskeeles Python tingimust kontrollida. Olulisel kohal on mõistmine, et tingimus on tõeväärtuse tüüpi – vastuseks saab olla ainult jah või ei.

Uute teadmiste kinnistamiseks lõimitakse need juba tuttavate teadmistega – varasematest praktikumidest andmetüübid, muutujad, aritmeetika tehted kasutaja sisend ja väljund ning matemaatikast naturaalarvud, võrduse kontroll, ringi läbimõõt ja pindala. Iga praktikumi materjalis ette nähtud ülesanne on illustreeritud visuaalse näitega korrektsest väljundist.

Antud praktikumi lisaülesanne sunnib taaskord õpilasi iseseisvalt mõtlema ja interneti abi kasutama. Ülesandeks on lihtsustatud versioon paljudes programmeerimiskursustes tuntud ülesandest „*FizzBuzz*“ [25]. See on põnev nuputamisülesanne, mis paneb proovile õpilaste arusaama tingimuslausete kontrollimise järjekorrast.

### 4.2.5. Praktikumid 5 ja 6

Viiendas praktikumis (Lisa VIII) süvendatakse õpilaste teadmisi tingimuslausest ja antakse sissejuhatus järjendi teemasse. Õpilasi õpetatakse kasutama algelisel tasemel tõeväärtustabelit – hindama, millistel tingimustel on tulemus tõene või väär.

Tõeväärtustabeli baasil on üles ehitatud ülesanne 1. Antud ülesandes tuleb matemaatikas varasemalt õpitu abil teha kindlaks, kas kolmnurk saab eksisteerida. Siiski ühtlustamaks kõigi taset, on vihjena välja toodud, mis tingimusel kolmnurk eksisteerida saab. See ülesanne paneb lisaks õpilased olema tähelepanelikud erinevate võimalike sisendikombinatsioonide suhtes.

Sissejuhatus järjenditesse antakse matemaatikas õpitud loetelu ja hulkade teema põhjal. 7. klassis õpitakse selgeks erinevad arvuhulgad, mis teataval määral ühtivad neile tuttavate andmetüüpidega. Eraldi tabelina on välja toodud enimkasutatavad tegevused, mida järjenditega teha saab. Nende illustreerimiseks on välja toodud avaldised koos tegevuse selgitusega ja tagastatava väärtusega. Neid kasutades sooritavad õpilased ülesande, mis on oma olemuselt seotud andmeteadusega. Ülesandes tuleb järjenditest programmide koostamise abil informatsiooni kätte saada. Selleks, et õpilastel oleks lihtsam enda programmi õigsust kontrollida, on iga osa ülesandest illustreeritud korrektse väljundiga.

Kuues praktikum (Lisa IX) on süvendamaks teadmisi järjendite kohta ja tuletatakse meelde enamlevinud järjendite funktsioonid. Seejärel lõpetatakse möödunud praktikumis pooleli jäänud viimane ülesanne.

### 4.2.6. Praktikum 7

Seitsmendas praktikumis (Lisa X) kesksel kohal on arusaam korduvatest tegevustest. Tsükliga tegutsemine kipub algajatele programmeerijatele enim raskust valmistama. Selle leevendamiseks on praktikumi materjalide alguses tehtud põhjalik ülevaade tsükli olemusest koos näitega, kus käiakse üksikasjalikult läbi ühe kindla tsükli toimimine. Selleks, et tsüklit realiseerida, selgitatakse ära, kuidas võrreldava muutuja väärtust muuta tsükli sees, et see ei jääks lõpmatult tegutsema. Samuti selgitatakse, kuidas võib tekkida olukord, kus tsükkel ei täida oma sisu kunagi.

Tsükli olemusest arusaamise kontrollimiseks on antud ülesanne 2. Ülesanne on võetud gümnaasiumi valikkursuse „Programmeerimine“ digiõpikust [26]. Selle inspiratsioonil lisati juurde veel mõned alternatiivid sarnases stiilis. See ülesanne annab õpilastele väärtusliku kogemuse mõelda ise läbi tsükli käitumine. Antud ülesande näol on tegemist väga lihtsate ja lühikeste programmijuppidega, mis on hoomatavad algajatele.

Ülesannetes 7, 8 ja 9 tuleb väljastada ekraanile kindlad arvud antud vahemikus. Esialgu tuleb neil lihtsalt väljastada kõik arvud selleks, et saada üldine programmi struktuur valmis. Seejärel järgmised ülesanded on juba variatsioonid sellest. Need ülesanded õpetavad õpilasi manipuleerima tsükli tööd vastavalt vajadusele.

Praktikumi lõpus saavad õpilased taaskord meelde tuletada esimeses praktikumis kasutatud Turtle moodulit, kuid sel korral tuleb joonistada korrapäraseid kujundeid tsükli abil. Selle jaoks saavad õpilased ette võtta varasemalt tehtud programmijupid ja uurida, kuidas teha seda efektiivsemaks. Ülesandeks on uurida, millised tegevused korduvad, ja kuidas neid tsükli abil üldistada saaks. Kiirematele on lisaülesandeks antud nuputada välja, kuidas saaks tsükli ja Turtle mooduli abil spiraali joonistada.

### 4.2.7. Praktikumid 8 ja 9

Kaheksandas ja üheksandas praktikumis (Lisa XI) keskendutakse tsükli teema kordamisele, kuid sel korral juba järjest keerulisemate programmide koostamise näol. Selle praktikumi peamine väljund on see, et õpilased suudaksid iseseivalt võimalikult palju kordustega seotud ülesandeid lahendada.

Esimesed kaks ülesannet on lihtsamad, et korrata eelmises praktikumis õpitut. Ülesanne 3 on võetud kursuselt „Programmeerimisest maalähedaselt“ [27], kus muudetud on vaid natuke sõnastust. See ülesanne on õpilaste esimene kokkupuude mahukama tsükli ülesandega. Ülesanne sunnib õpilased korduvalt läbi mõtlema juhendi ja tsükli tegevuse, rakendama pea kõiki õpitud teadmisi ja kasutama programmeerimiskeskkondades antud koodi silumise funktsiooni.

Ülesanne 4 on koostatud Collatzi hüpoteesi [28] baasil, mis on üks kuulsamaid lahendamata probleeme matemaatikas. See ülesanne paneb õpilased keskenduma ülesande püstitusele ja

lahendama ülesandeid osadena. See on hea jõukohane nuputamisülesanne III kooliastme õpilastele.

### 4.2.8. Praktikum 10 ja 11

Kümnendas ja üheteistkümnendas praktikumis (Lisa XII) võetakse kasutusse kõik kursuse vältel omandatud teadmised ja rakendatakse neid ülesannetes, mida saab pidada projektivääriliseteks. Selles praktikumis on oluline, et õpilased suhtleksid omavahel ja arutaksid ülesannete võimalikke lahendusi.

Kahe antud ülesande näol on tegemist kõigile tuttavate mängudega – numbri arvamise mäng ja mäng „kivi, paber, käärid“. Numbri arvamise mäng tutvustab õpilastele kontseptsiooni, kus kasutaja suhtleb arvutiga selliselt, et saab sellelt pidevalt tagasisidet. Sarnane kontseptsioon on kasutusel paljudes olukordades, näiteks olukord, kus kasutaja peab sisestama salasõna ja süsteem annab teada, kas tegu on õige või vale parooliga.

Mängu „kivi, paber, käärid“ koostamisel saavad õpilased iseseisvalt õpetaja juhendamisel tutvuda suvalise numbri genereerimisega, mis on tõenäoliselt kasulik teadmine neile edaspidistes praktikumides tehtava projekti raames. Praktikumi eduka läbimise korral on õpilased valmis alustama iseseisva projekti koostamisega.

### 4.2.9. Praktikumid 12-16

Praktikumides 12-16 (Lisa XIII) on kasutatud projektipõhise õppe printsiipe. Õpilased peavad viie praktikumi jooksul jõudma antud tingimustel projekti ideest valmis programmini. Selle jaoks tuleb kasutada kõiki praktikumides omandatud teadmisi. Selleks, et õpilased saaks enda loovust kasutada, välditakse ülesannete ette andmist, kuid ideede puudumisel pakuti välja nimekiri võimalikest projektidest.

Praktikumis 12 peavad õpilased etteantud kriteeriumite raames mõtlema välja tehtava projekti. Selle jaoks on tarvis juhendajale esitada detailne plaan sellest, mida nende programm sisaldama peaks. See sunnib õpilasi üksikasjalikult läbi mõtlema iga komponendi oma programmi tegevuses. Vajadusel suunab neid õpetaja mõtlema, kuidas enda plaani täiendada.

Praktikumides 13-16 realiseerivad õpilased enda projekti. Juhendajalt abi küsimine on võimalik, kuid rangelt soovituslik on olemasolevate ressursside abil ise vastuse leidmine. Iga praktikumi lõpus peavad õpilased näitama juba tehtud tööd ja selgitama plaani järgmiseks praktikumiks.

### 4.2.10. Praktikum 17

Kursuse lõpetab praktikum 17 teadmiste kontroll. Teadmiste kontroll toimub õpilastele testi vormis, kuid testi sooritus ei määra kursusest läbi saamist. Testi eesmärk on õpetajal ja õpilasel saada informatsiooni kursuse efektiivsusest.

Testi koostamisel oli oluline kontrollida kõiki praktikumides 1-11 saadud teadmisi (vt peatükk „3.2. Kursuse ja õppematerjalide koostamise protsess“). Test on jaotatud kaheks osaks: test teooria peale ja programmi koostamine.

Testis teooria peale (Lisa XIV) on 10 küsimust, millest viis on valikvastustega ja ülejäänud avatud vastustega. Testiga üritatakse aru saada, millisel määral on õpilased suutelised arvuti abita mõistma ja analüüsima etteantud programmi tööd. Test toimub paberil suletud materjalidega, kus õpilastele on ette antud 10 erinevat lühikest programmi. Seejuures ülesanded on progresseeruva raskusega ja nende lahendamiseks on aega 30 minutit.

Algtasemel informaatikas on oluliseks väljundiks praktiline programmide koostamise oskus. Programmi koostamise ülesandega (Lisa XV) üritatakse mõista, kas ja millisel määral õpilased on omandanud kursuse vältel õpitud teemad. See osa testist toimub arvutis, kus õpilased võivad lahendades kasutada enda poolt varem koostatud programme, kuid omavaheline suhtlus on rangelt keelatud. Testi koostamisel võeti arvesse, et õpilased oleksid sunnitud kasutama võimalikult palju erinevaid omandatud teadmisi ja selle testiosa lahendamiseks on aega 60 minutit.

## 5. Tagasiside analüüs

Selles peatükis antakse ülevaade õpilastelt kogutud tagasisidest. Kursuse vältel koguti igas tunnis kõigilt kohalolijatelt andmeid praktikumide korralduse ning ülesannete ja teemade keerulisuse kohta. Lisaks viidi viimases tunnis läbi põhjalikum avatud vastustega küsimustik, kus õpilased said pikemalt enda arvamused kursuse ja programmeerimise kohta kirja panna.

### 5.1. Osalejate taust

Peatükis „3.1. Valim“ anti ülevaade kursuse läbinud õpilastest. Joonis 1 väljendab õpilaste kuulumist nii klassidesse kui temporühmadesse. Sealt selgub, et huvi programmeerimise vastu ei ole üksnes ainult reaal- ja loodusainetes tugevamatel, vaid ka keskmise tasemega III kooliastme õpilastel.

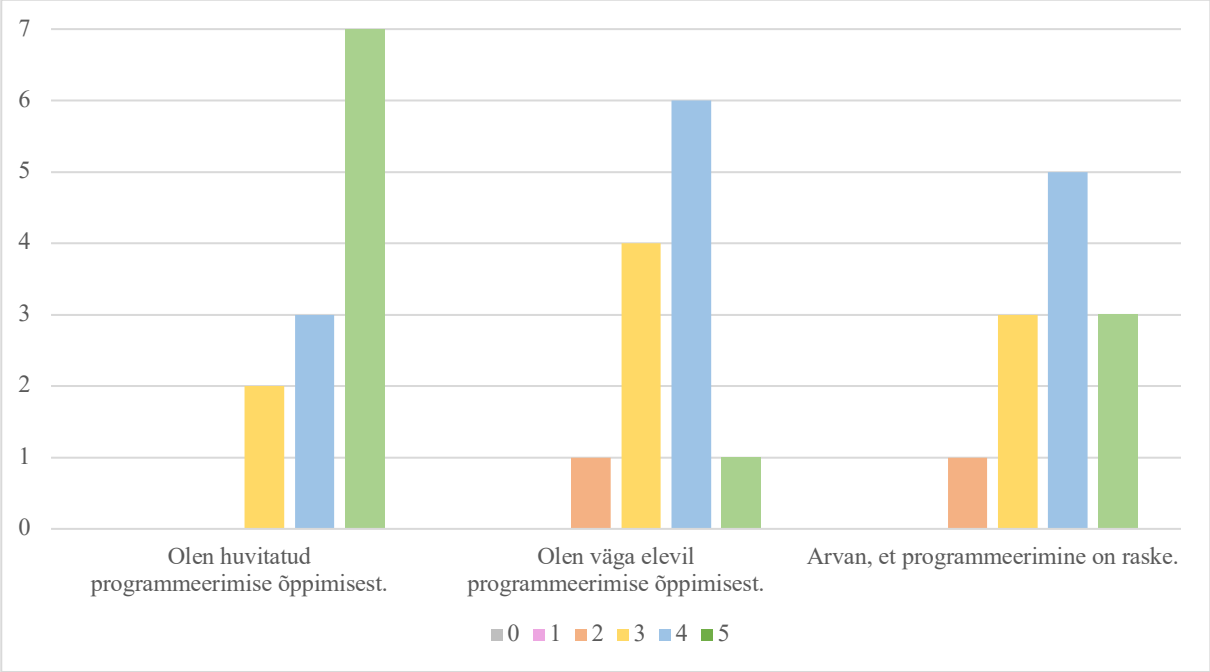
Kuigi antud kursusest ei võtnud osa temporühma „aeglane“ õpilased, ei välista see nende huvi programmeerimise vastu. Õpilased said valida 14 erineva praktikumi vahel, kus kohad osalemiseks olid limiteeritud ja praktikumikohtade täitumine toimus ajalise kiiruse alusel. See tähendab, et on võimalik olukord, kus mõni temporühma „aeglane“ õpilane soovis programmeerimise praktikumist osa võtta, ent ei jõudnud piisvalt kiiresti registreerida või eksisteeris valikus praktikum, millest õpilane soovis rohkem osa võtta.

Valimi tulemuste analüüsimisel on oluline teada nende varasemast kokkupuutest programmeerimisega. Jooniselt 2 selgub, et enamus õpilasi oli enne kursusega alustamist proovinud õppida programmeerimist iseseisvalt. See annab meile teada, et III kooliastme õpilased on huvitatud programmeerimise õppimisest ja kasutavad võimalust seda teha – toimugu see klassiruumis või iseseisvalt.

Õpilaste harjumused ja suhtumised mõjutavad seda, kui sujuvalt kulgevad nende õpingud. Enne kursusega alustamist tehti kindlaks, milline on keskmise kursusel osaleva õpilase profiil. Seda tehti peatükis „3.3. Tagasiside kogumine ja andmeanalüüs“ väljatoodud viisil. Tulemused kajastuvad joonistel 3 ja 4.

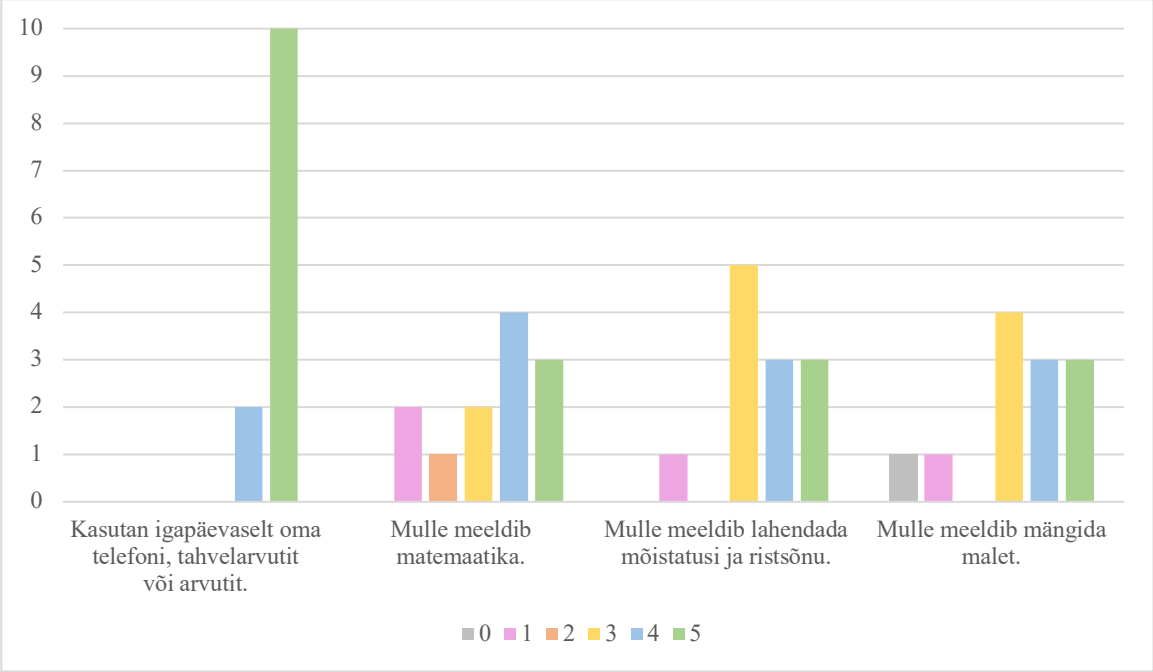
Jooniselt 3 on näha, et õpilased on programmeerimise õppimise suhtes positiivselt meelestatud. Keskmise kursusel osaleva õpilase huvi programmeerimise vastu viie punkti skaalal oli ~4,4. Seejuures hindas keskmine õpilane oma elevust programmeerimise õppimise osas samal

skaalal hindega ~3,6. Kuigi antud tulemus jääb pigem positiivselt meelestatuse poole, on see oluliselt madalam kui hinnang huvi osas. Nende kahe tulemuse vahet võib selgitada asjaolu, et üldiselt ei olda elevil mõttest õppida. Seda toetab jooniselt 3 välja tulnud hinnang, et õpilased arvavad, et programmeerimine on pigem raske. Keskmine õpilane pidas viie punkti skaalal programmeerimise keerulisuseks 4,8. Saab öelda, et kuigi III kooliastme õpilased selles praktikumis peavad programmeerimist raskeks, siis on nad valmis ja soovivad seda õppida.




Joonis 3. Õpilaste meelestatus programmeerimise õppimise suhtes.

Jooniselt 4 on näha, et õpilased kasutavad oma telefoni, tahvelarvutit või arvutit igapäevaselt. Seega ei tohiks arvuti kasutamine osutuda neile keeruliseks. Edasi hindasid nad oma suhtumist matemaatikasse väitega „Mulle meeldib matemaatika.“. Keskmine õpilane on pigem neutraalne oma suhtumises matemaatikasse, hinnates seda viie punkti skaalal hindega ~3,4. Küll aga üldiselt koolivälised tegevused nagu male mängimine ning mõistatuste ja ristsõnade lahendamine, meeldisid õpilastele keskmiselt natuke rohkem. Keskmise arvutamisest jäeti välja õpilased, kes valisid vastuseks „0“. Seega keskmine õpilane hindas oma suhtumist viie punkti skaalal nii mõistatuste ja ristsõnade lahendamisse kui ka male mängimisse hindega ~3,6. Need tulemused näitavad, et huvi programmeerimise vastu ei sõltu tingimata sellest, kas ja kui palju õpilasele meeldivad muud loogilist mõtlemist nõudvad tegevused.

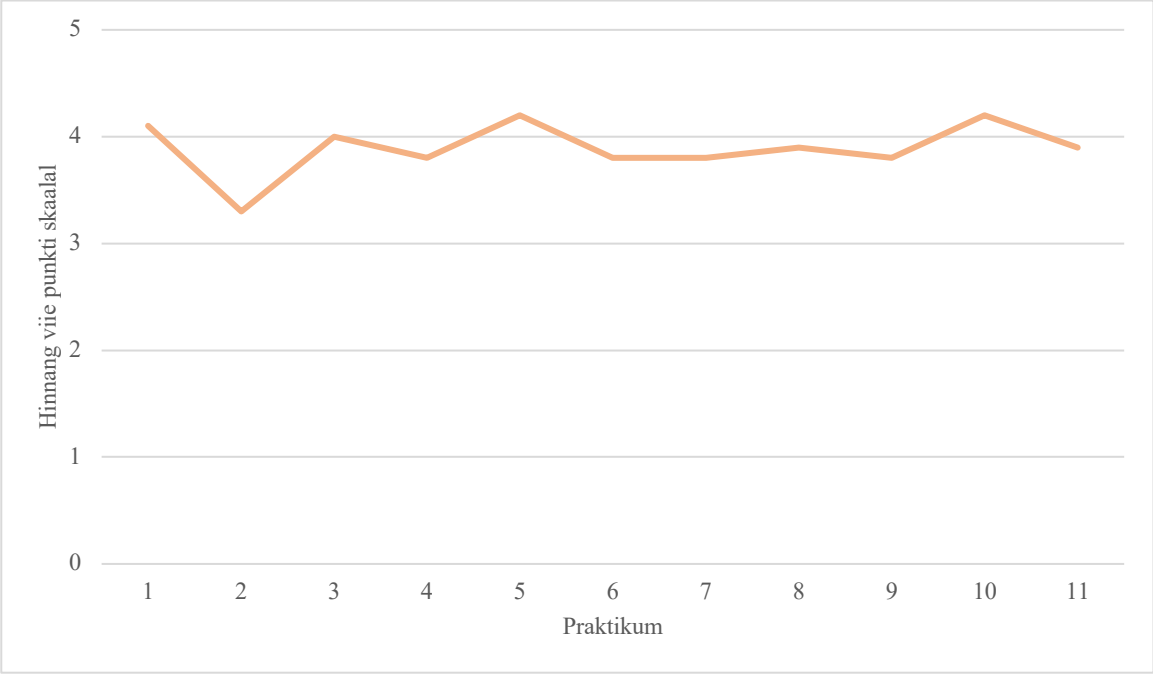



Joonis 4. Õpilaste harjumused ja suhtumised.

## 5.2. Tagasiside praktikumide korralduse ja ülesannete keerulisuse kohta

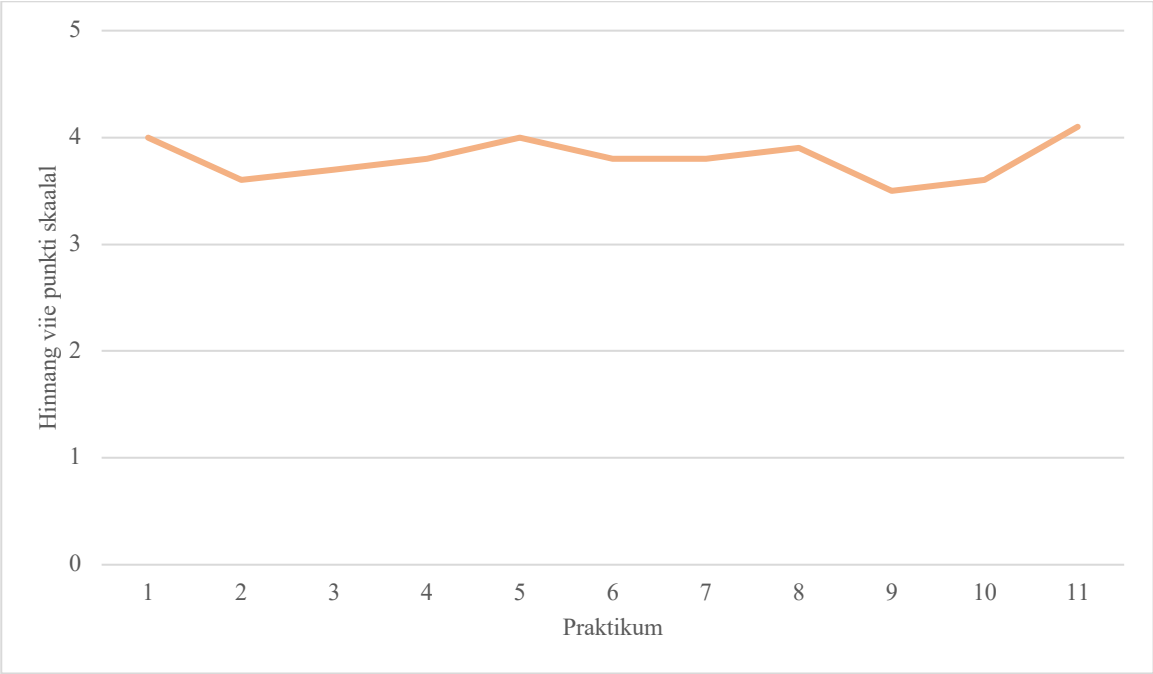
Igas praktikumis, kus omandati programmeerimise baasteadmisi, koguti kohalolijatelt tagasisidet tundide korralduse ja ülesannete keerulisuse kohta. Teemade õppimine toimus praktikumides 1-11. Peale igat praktikumi vastasid õpilased peatükis „3.3. Tagasiside kogumine ja andmeanalüüs“ välja toodud väidetele.

Jooniselt 5 on näha, et üldiselt õpilastele tunnid meeldisid. Keskmine hinnang tundide meeldimise kohta viie punkti skaalal oli ~3,9. Enim meeldisid õpilastele esimene ja viies praktikum. Esimeses praktikumis tegeleti sissejuhatusega programmeerimisse ja viiendas tingimuslausega. Esimeses praktikumis said õpilased kilpkonnagraafika abil lahendada ülesandeid ja teha omaloomingut. Viiendas said õpilased esimest korda luua keerukamaid programme, kus programmi käik sõltus kasutaja sisendist. Küll aga kõige vähem meeldis õpilastele teine praktikum. Seda võib selgitada asjaolu, et teises praktikumis tegeleti peamiselt teooria õppimisega ja ise programmide loomist oli vähe. Kuigi õpilastele meeldis see tund kõige vähem, on see just üks olulisemaid praktikume, et panna alustala nende edasistele õpingutele kursuse vältel. Olenemata sellest, et antud praktikumi hinnang on silmnähtavalt kõige nõrgem, ei tasu selle sisu muuta – tänu selles tunnis saadud teadmistele on õpilastel võimalik tunda end järgmiste teemade omandamisel enesekindlamalt.



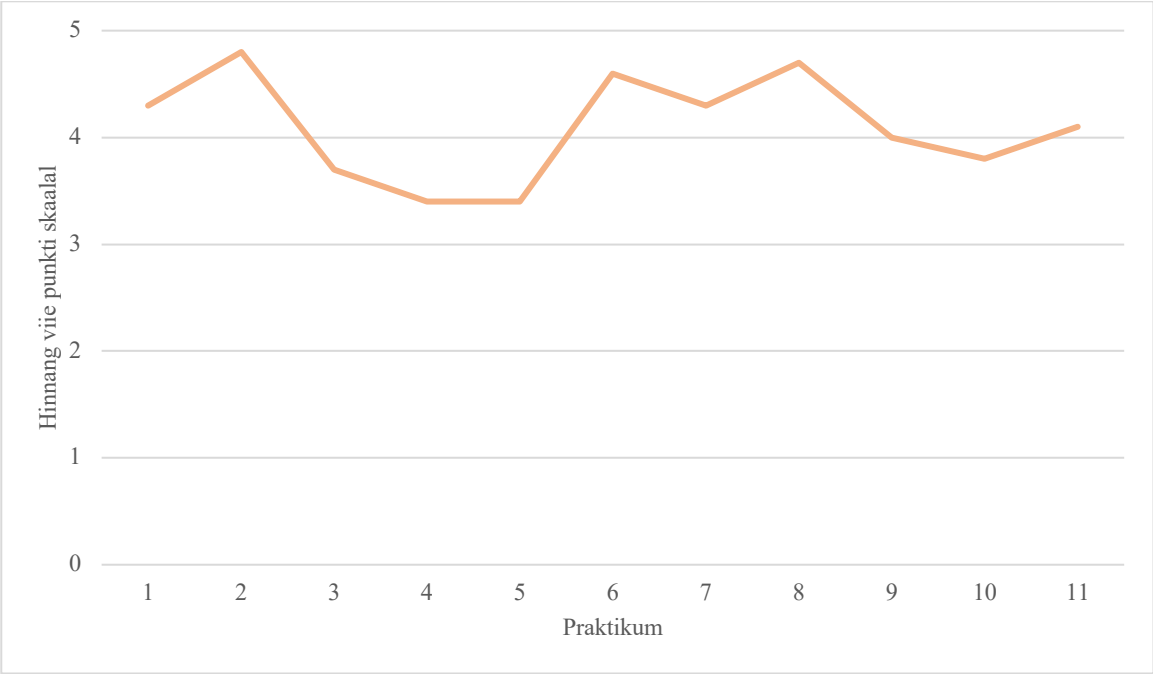

Joonis 5. Keskmise õpilase hinnang tunni meeldimise kohta.

Jooniselt 6 on näha, et praktikumides antud ülesanded on kolmanda kooliastme õpilastele jõukohased. Keskmine hinnang praktikumis tehtud ülesannete jõukohasuse kohta viie punkti skaalal oli 3,8 – suurt kõrvalekallet keskmisest läbivalt ei olnud. Praktikumides 8, 9, 10 ja 11 tegeleti tsüklite õppimisega. Jooniselt on näha, et sellel teemal lahendatud ülesanded olid õpilastele algselt kõige keerulisemad. Huvitav tähelepanek on, et kui praktikum 8 sai üsna kõrge hinnangu jõukohasuse osas, siis praktikumi 9 ülesanded samal teemal olid õpilastele märksa keerulisemad – pärast seda iga praktikumiga paistsid ülesanded muutuvat taaskord üha jõukohasemaks. Erinevus praktikumide 8 ja 9 ülesannete jõukohasuse osas tekib tõenäoliselt sellest, et teema sissejuhatuses üldiselt tehakse lihtsamaid ülesandeid, et saada selgeks üldised põhimõtted. Seejärel liigutakse aina raskemate ülesannete juurde. Küll aga lõpuks harjuvad õpilased keerulisema raskusastmega ära ja see selgitaks ka tõusu jõukohasuse hinnangus praktikumides 10 ja 11.



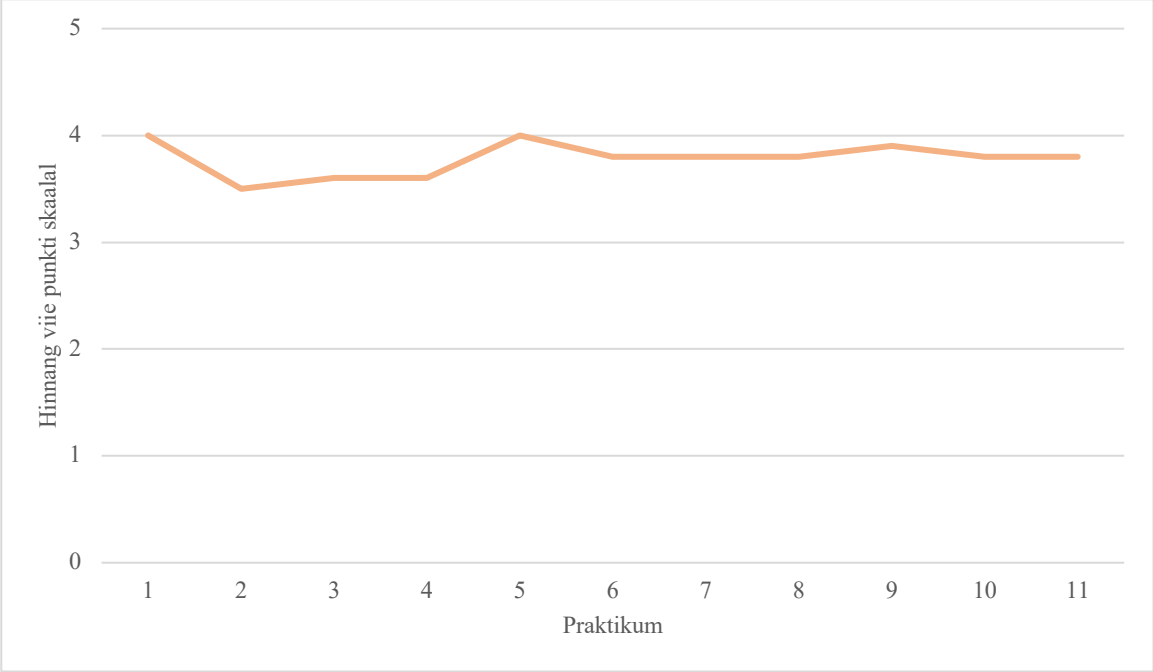

Joonis 6. Keskmise õpilase hinnang praktikumis tehtud ülesannete jõukohasuse kohta.

Jooniselt 7 on näha, et praktikumis ettenähtud ülesannete valmisjõudmine varieerus palju. Keskmine hinnang sellele viie punkti skaalal oli 4,1. Esimestes sissejuhatavates praktikumides jõudis keskmine õpilane peaaegu kõik ülesanded valmis, ent järgmises kolmes praktikumis jäi ülesannete maht liiga suureks. Seda täheldades tehti alates kuuendast praktikumist otsus panna töölehtedele väiksem kogus põhiülesandeid ja lisaks tärniga ülesandeid, mida saavad teha need õpilased, kes on teistest kiiremad. Jooniselt 7 on näha, et uus töömaht oli keskmisele õpilasele paras – kui kolmandast viienda praktikumini oli ülesannete valmisjõudmise hinnang keskmisel õpilasel 3,5, siis peale muudatusi praktikumimaterjali ülesehituses oli see 4,3. Korrektsioonide positiivse mõju tõttu muudeti hiljem ka praktikumide 3, 4 ja 5 ülesehitus sarnaseks teistega.




Joonis 7. Keskmise õpilase hinnang praktikumis ettenähtud ülesannete valmisjõudmise kohta.

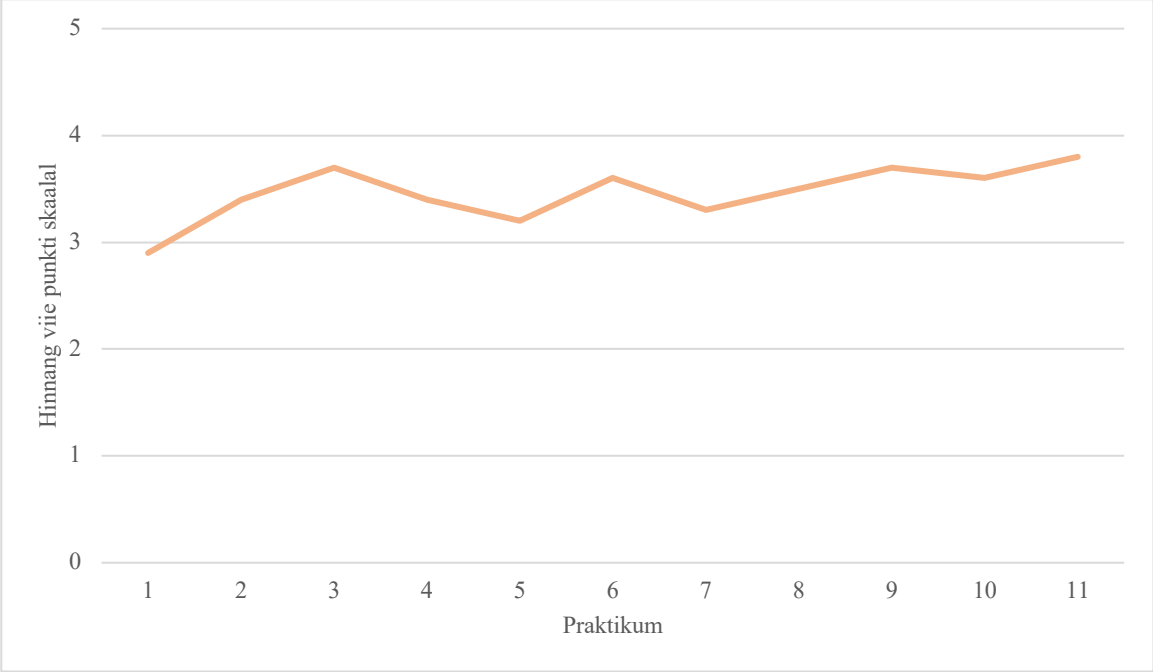
Jooniselt 8 on näha, et üldiselt oli õpilastel praktikumides läbivalt piisavalt meeldiv, et keskmine inimene hindas järgmise praktikumi toimumise ootamist viie punkti skaalal hindega ~3,8. Siin tuleb arvestada, et õppimine on kurnav tegevus, mistõttu hinnati seda küsimust palju arvuga „3“, mis väljendas neutraalsust. Seejuures negatiivsed vastused läbivalt puudusid.




Joonis 8. Keskmise õpilase hinnang sellele, kui väga ootab järgmise praktikumi toimumist.

Jooniselt 9 on näha, et õpilased pidasid programmeerimist mõõdukalt keeruliseks, hinnates seda viie punkti skaalal skooriga ~3,5. On märgata trendi, et peale igat praktikumi peeti programmeerimist üha keerulisemaks. Kui peale esimest praktikumi hinnati programmeerimise keerulisust keskmisest madalama hindega, siis viimase praktikumi lõpuks hinnati programmeerimist viie punkti skaalal pea punkti võrra keerulisemaks. See on tavaline nähtus paljude erinevate kursuste puhul – kursuse alguses on üldiselt lihtsamad sissejuhatavad teemad, kuid mida aeg edasi, seda raskemaid teemasid läbitakse.

Enne kursuse algust paluti õpilastel hinnata viie punkti skaalal, kui keeruliseks nad programmeerimisest peavad (joonis 3) - keskmiseks hinnanguks oli 4,8. Kursuse vältel pidasid õpilased programmeerimist läbivalt kergemaks, kui nad algselt arvasid seda olevat. Sellest saab järeldada, et vähene kokkupuude või selle täielik puudumine tekitab õpilastele tunde, et programmeerimine on keerulisem, kui see päriselt on. Läbi meedia saadud idee programmeerimisest võib panna õpilased arvama, et tegemist on müstiliste inimesele loetamatute keeltega. Vähene kokkupuude programmeerimisega võib tekitada tingimused, kus inimesed endale teadmata seostavad programmeerimist vaid masinkoodiga.

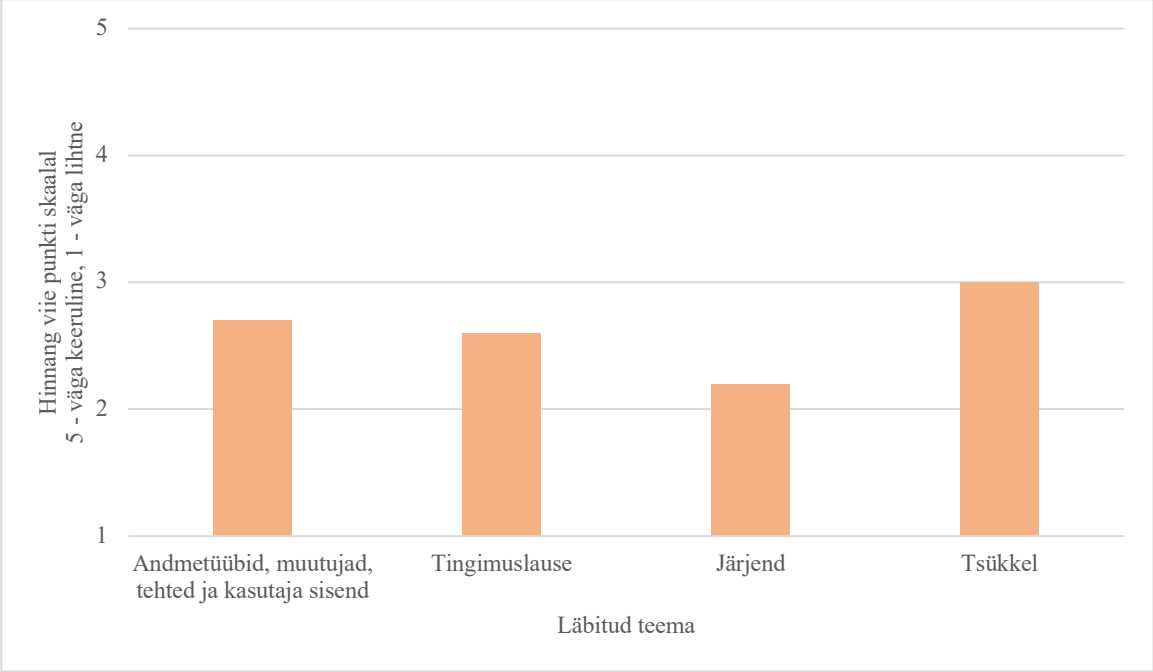



Joonis 9. Keskmise õpilase hinnang programmeerimise raskuse kohta pärast iga praktikumi.

## 5.3. Kursuse vältel kogutud tagasiside läbitud teemade keerulisuse kohta

Peale iga teema täielikku läbimist hindasid õpilased peatükis „3.3. Tagasiside kogumine ja andmeanalüüs“ välja toodud viisil teemade raskusastet. Tulemused saadud tagasiside osas on esitatud joonisel 10.

Jooniselt 10 on näha, et keskmine õpilane antud kursusel ei pidanud ühtegi läbitavat teemat keeruliseks. Keskmiseks hinnanguks teemade keerulisuse osas anti ~2,6. Kõige raskemaks teemaks peeti tsükleid. Seda just seepärast, et tsüklite korral on väära lõpptulemuse korral keeruline kindlaks teha, kus viga tekib. Kursuse materjale koostades eeldati seda tulemust ja seepärast näidati õpilastele juba esimestes praktikumides silumise (ingl *debugging*) võimalust. Koodi silumine on hea viis saada aru, kuidas programm päriselt töötab – see annab võimaluse käia koodi läbi ridahaaval, saades info, mis väärtusi programmi vältel erinevad muutujad omistavad.




Joonis 10. Keskmise õpilase hinnang läbitud teemade keerulisuse kohta.

## 5.4. Kursust lõpetava teadmise kontrolli tulemused

Sama oluline, nagu on õpilaste enda suhtumine ja tunded programmeerimise õppimisel, on oluline kontrollida, kas ja millisel määral said õpitud teemad omandatud. Selleks viidi kursuse viimases tunnis läbi teadmiste kontroll. See koosnes kahest osast: test teooria peale ja programmi koostamine.

### 5.4.1. Kursuse lõputesti tulemused

Test teooria peale koosnes kümnest programmijupist (Lisa XI), mille tulemusi õpilased ennustama pidid. Ülesanded olid progresseeruva raskusastmega ja tabelis 1 on välja toodud iga küsimuse kohta tulemused – testi tegid kõik 12 õpilast, kes kursusel olid.

Tabel 1. Teoreetilises testis ülesannetes õigesti lahendanute arv iga ülesande kohta.

| **Ülesande number** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Õigesti lahendanute arv** | 12 | 12 | 12 | 12 | 12 | 10 | 11 | 9 | 8 | 8 |

Kõigi õpilaste keskmine tulemus oli 88,3% standardhälbega 14,7 - kõrgeim tulemus oli 100% ja madalaim 50%. Ülesannetes, kus olid ette antud vastusevariandid, valisid kõik õpilased õige

vastuse. Kui avatud vastusega ülesandes eksiti, siis üldiselt oli see siiski õigele vastusele lähedal. Rohkem vigu tuli sisse tsüklite läbimisel, kuid kõige keerulisemaks osutusid ülesanded, kus oli omavahel kombineeritud tsüklite läbimist järjendiga.

Teoreetilise testi tulemuste põhjal saab öelda, et kursus oli antud grupi näol edukas. Teemade läbimine vastavalt õpilaste enda tempole tagas kursuse lõppedes olukorra, kus keskmine õpilane oli need teemad väga hästi omandanud. Lisaks toetas seda asjaolu, et grupis oli kõigest 12 õpilast – selliselt oli juhendajal võimalus individuaalselt teemad õpilastele lahti seletada just selliselt, et nemad sellest aru saaksid.

### 5.4.2. Programmi koostamine

Programmi koostamine toimus etteantud ülesande (Lisa XII) alusel. Täpset numbrilist tulemust selle eest õpilastele ei antud. Küll aga anti neile verbaalselt individuaalset tagasisidet.

Ülesannete lahendused olid väga omanäolised, kuid üldiselt saadi lahendamisega hästi hakkama. Kuigi tsükkel õpilaste jaoks oli kõige raskem teema (vt joonis 10), sai iga õpilane tsükli abil järjendi läbimisega hakkama. Arvuliste väärtuste leidmine järjendist probleemiks ei osutunud – igas lahenduses leidus kontroll selle jaoks, et teha kindlaks, kas keskmise arvutamisel saab järjendi elementi kasutada.

Järjendist keskmise leidmine paistis õpilaste jaoks lihtne. Seda ilmselt seetõttu, et praktikumide materjalides oli sarnase sisuga ülesandeid. Esimesed vead tekkisid, kui õpilased pidid keskmisest 15% suurema arvu leidma. Üldiselt saadi sellega hästi hakkama, kuid paar olid selle üldse tegemata jätnud. Ka ümardamine ei valmistanud õpilastele palju probleeme – vaid üks õpilane oli jätnud selle tegemata. Teksti koos leitud väärtusega ekraanile väljastamine tuli kõigil hästi välja.

Kokkuvõtteks võib öelda, et kursus oli programmi koostamise näol edukas. Iga õpilane sai programmi koostamisega hakkama ja esines vaid üksikuid õpilasi, kes tegi mõne vea. Arvestades kõiki omandatud teadmisi, jäid need vead suures pildis väheolulisteks ja võisid olla tähelepanematusest.

## 5.5. Avatud tagasiside kursuse lõpus

Üldiselt arvasid õpilased, et neil läks kursusel hästi. Enim toodi välja, et neil oleks paremini läinud, kui oleksid esimese korraga kohe kuulanud ja teadmised meelde jätnud või kõrval oleva õpilasega vähem jutustanud. Kuna tegemist oli nende jaoks uue ainega, siis toodi välja, et alguses oli pisut keeruline harjuda, kuid lõpus said kursuse struktuuri tõttu paremini aru, kuidas efektiivsem õppida oleks. Samuti toodi välja, et kursus oleks paremini läinud, kui oleks suutnud kiiremini ära otsustada, mis projekti teha.

Kõige rohkem meeldisid õpilastele Turtle'i abil joonistamine, tingimuslaused, iseenda projektiga tegelemine ja järjendid. Samas toodi välja, et kõige vähem tulid välja keerulisemad ülesanded, kus pidi kasutama tsüklit. Samas nentisid mõned õpilased, et nende jaoks oli enda projekti tegemine siiski keeruline.

Õpilased tõid välja, et nautisid enim kursusel seda, et kursuse töö toimus arvutis. Enamus õpilasi vastas, et nende jaoks oli nauditavad just ülesanded, mida nad ei pidanud ülemäära keeruliseks. Nende sõnul andis see suure rahulolutunde, kui kirjutatud koodijupp korrektselt töötas. Samuti toodi meeldivana välja joonistamine ning sõpradega koos projekti tegemine.

Oma tulevikuplaanidega seoses programmeerimisega ütlesid õpilased palju positiivset. All on toodud väljavõtted kõigi õpilaste vastusest (tekstis kirjavigade esinemisel on töö autor need vastavalt parandanud).

- „Ei ole kindel.".
- „Jah!".
- „Jah.".
- „Ma veel ei tea, kuid ma ei kavatse programmeerimist vältida.".
- „Ma arvan, et võin veel kokku puutuda, aga mul ei ole selle jaoks veel otseselt plaani.".
- „Plaanin gümnaasiumis õppida programmeerimist ning loodetavasti saada ka sel alal töö.".
- „Võib-olla, aga ei ole kindel, kas kasutan siis Pythonit või mingit muud programmeerimiskeelt, näiteks Javat.".
- „Ma plaanin tulevikus programmeerimisega kokku puutuda. Praegusel hetkel on mul plaanis sellest oma tuleviku töö teha.".

- „Jah. Ma tahan tulevikus programmeerijaks saada ja hakkan varsti iseseisvalt õppima. Ma veel ei tea kuidas, aga proovin ikka.“.
- „Ma arvan, et võib-olla plaanin minna isegi programmeerimise poole kui võimalik. Programmeerimisega saab teha palju erinevaid asju.“.
- „Ma arvan, et see oleks tore. Kuna praktikum meeldis, ei näe ma mingit põhjust, miks mitte. Samas, hetkel mul sellega seoses veel mingit otsest plaani ei ole.“.
- „Ma arvan, et puutun veel kokku, kuna programmeerimine on nii suur osa meie praegusest maailmast ja aina olulisemaks see tulevikus läheb.“.

Sellest tagasisidest saab järeldada, et kursus oli edukas. Kursus täitis oma funktsiooni, üritades olla üks positiivseid esimesi kogemusi programmeerimisega õpilastele III kooliastmes. Kuigi kursuse koostamisel ei olnud eraldi eesmärk, et õpilased läheksid seda edasi õppima, siis mitme õpilase näol tekkis kursuse läbimisel suurem plaan seoses informaatikaga. See näitab, et kokkupuude informaatikaga juba III kooliastmes on oluline, et õpilased saaksid teha informeeritud otsuse, kuidas oma tulevikku luua. Samamoodi nagu paljud erinevad õppeained, peaks ka programmeerimise algkursus olema põhikooli õppekavas, et tänases maailmas õpilased saaksid avastada endale huvipakkuvaid alasid.

## 6. Kokkuvõte

Bakalaureusetöö eesmärk oli töötada välja õppematerjalid programmeerimise õpetamiseks III kooliastme õpilastele. Õppekava koosneb seitsmeteistkümnest 90-minutilisest tunnist, millest igaüks oli mõeldud õpilastele väljakutseid pakkuvaks, kuid nauditavaks õppimiskogemuseks. Kogu arendusprotsessi vältel andsid õpilased väärtuslikku tagasisidet kursuse materjalide kohta.

Küsitluse tulemused näitasid, et selle töö raames välja töötatud programmeerimiskursus võeti õpilaste poolt hästi vastu. Selle alusel saab väita, et õpilastele kursus meeldis. Lisaks sellele saab öelda, et kursus on III kooliastme õpilastele jõukohane. Samuti tuli töö käigus välja, et kursus vähendas õpilaste ettekujutust sellest, et programmeerimine on raske. Tegemist on väga olulise järeldusega, kuna paljusid õpilasi sageli alguses hirmutab programmeerimine.

Kursust lõpetava testi tulemused näitasid, et õpilased omandasid hea arusaamise kursusel käsitletavatest teemadest. Lisaks väljendasid paljud õpilased huvi tulevikus programmeerimisega tegelemise vastu, rõhutades selle kursuse võimalikku mõju nende tulevikuplaanidele ja karjääripüüdlustele.

Võib öelda, et lõputöö on näidanud informaatikaalase hariduse olulisust ja jõukohasust III kooliastme õpilaste jaoks. Analüüsides õpilaste tagasisidet õpikogemuse kohta ja kursust lõpetava testi tulemust, saab väita, et selle töö raames välja töötatud programmeerimiskursus sobib III kooliastme õpilastele. See lõputöö rõhutab informaatika tutvustamise tähtsust noortele ja vajadust jätkata uute võimaluste uurimist, kuidas muuta programmeerimine õpilastele kättesaadavamaks ja kaasahaaravamaks.

Edasiarendusena saaks kirjutada juurde materjale teemade kohta, mida antud kursus ei käsitle. Vajadusel võib modifitseerida olemasolevaid materjale ja kursuse struktuuri paremaks. Õpilaste positiivse tagasiside põhjal saab öelda, et seda kursust võiks läbi viia veel erinevates koolides, saamaks rohkem andmepunkte selle sobivusest põhikooli õppekavasse.

## Viidatud kirjandus

[1] Kori, K., Mardo, K. Õppimine ja väljalangemine IKT erialade esimesel aastal Eesti kõrgkoolide näitel. *Eesti Haridusteaduste Ajakiri,* nr 4 (1), 239-267, 2017. https://ojs.utlib.ee/index.php/EHA/article/view/eha.2017.5.1.08/8464

[2] Altin, H., Rantsus, R. Why students fail to graduate ICT-related curricula at university level. *INTED2015 Proceedings*, 5364-5368, 2015.
https://sisu.ut.ee/sites/default/files/ikt/files/heilo_altin_inted.pdf

[3] Kori, K., Beldman, P., Tõnisson, E., Luik, P., Suviste, R., Siiman, L., Pedaste, M. IT oskuste arendamine Eesti koolides, 2019.
https://wise.com/documents/IT%20oskuste%20arendamine%20Eesti%20koolides.pdf
(03.01.2023)

[4] Wing, J. M. Computational thinking, Communications of the ACM, Vol 49, No. 3, pp 33-35, 2006. https://dl.acm.org/doi/fullHtml/10.1145/1118178.1118215

[5] Barr, V., Stephenson, C. Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? ACM Inroads, 2(1), 48-54, 2011. https://dl.acm.org/doi/pdf/10.1145/1929887.1929905

[6] Eckerdal, A., Berglund, A. What Does It Take to Leard 'Programming Thinking'?, 2005. https://dl.acm.org/doi/pdf/10.1145/1089786.1089799

[7] Grover, S., Pea, R. D. Computational Thinking in K-12 A Review of the State of the Field, 2013. https://www.researchgate.net/publication/258134754_Computational_Thinking_in_K-12_A_Review_of_the_State_of_the_Field

[8] Põhikooli riiklik õppekava. Riigiteataja I. Lisa 10 valikõppeaine „Informaatika". https://www.riigiteataja.ee/aktilisa/1140/7202/0024/1m%20lisa10.pdf (03.01.2023)

[9] Bosse, Y., Gerosa, M. A. Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage. SIGSOFT Softw. Eng. Notes 41, pp. 1-6, 2017. https://doi.org/10.1145/3011286.3011301

[10] Lahtinen, E., Ala-Mutka, K., Järvinen, H.-M. A Study of the Difficulties of Novice Programmers. https://dl.acm.org/doi/pdf/10.1145/1151954.1067453

[11] Duncan, C., Bell, T., Tanimoto, S. Should your 8-year-old Learn Coding? https://dl.acm.org/doi/pdf/10.1145/2670757.2670774

[12] Bawamohiddin, A. B., Razali, R. Problem-based Learning for Programming Education, 2017.

https://www.researchgate.net/publication/322135044_Problem-based_Learning_for_Programming_Education

[13] Nuutila, E., Törmä, S., Kinnunen, P., Malmi, L. Learning Programming with the PBL Method – Experiences on PBL Cases and Tutoring, 2008. https://link.springer.com/chapter/10.1007/978-3-540-77934-6_5

[14] Kokotsaki, D., Menzies, V., Wiggins, A. Project-based learning: A review of the literature, 2016. https://journals.sagepub.com/doi/pdf/10.1177/1365480216659733

[15] Öztürk, F., Özdemir, M., Özbaşi, D. Investigation on Project-Based Learning Method in Teaching Programming in terms of Academic Achievement, Cognitive Load and Behavior Change, 2021. https://files.eric.ed.gov/fulltext/EJ1301891.pdf

[16] Ling, F., Gong, S. Research on Project-Based Learning Python Programming Course, 2022. https://drpress.org/ojs/index.php/fcis/article/view/1884

[17] Teague, D. Pedagogy of introductory computer programming: a people-first approach, 2011. https://core.ac.uk/reader/10907450

[18] Faja, S. Evaluating Effectiveness of Pair Programming as a Teaching Tool in Programming Courses, 2014. https://files.eric.ed.gov/fulltext/EJ1140923.pdf

[19] TIOBE programmeerimiskeelte indeks. https://www.tiobe.com/tiobe-index/ (09.01.2023)

[20] danchikov, A., Zhaparov, M., Suliyev, R. Python to learn programming. *Journal of Physics: Conference Series*, Vol. 423, 012027, 2013. https://iopscience.iop.org/article/10.1088/1742-6596/423/1/012027/pdf

[21] Shein, E. Python for Beginners. *Communications of the ACM*, Vol. 58, No. 3, p. 19–21, 2015. https://dl.acm.org/doi/pdf/10.1145/2716560

[22] Annamaa, A. Thonny, a Python IDE for Learning Programming. *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, p. 343, 2015. https://dl.acm.org/doi/pdf/10.1145/2729094.2754849

[23] Silumine (*debug*). https://pydoc.pages.taltech.ee/workflow/debug.html (02.01.2023)

[24] Python Software Foundation. *Turtle graphics*. https://docs.python.org/3/library/turtle.html (06.05.2023)

[25] *FizzBuzz In Too Much Detail*, 2015. https://www.tomdalling.com/blog/software-design/fizzbuzz-in-too-much-detail/#:~:text=It%20was%20invented%20by%20Imran%20Ghory%2C%20and%20popularized%20by%20Jeff%20Atwood (10.08.2023)

[26] Tõnisson, E., Palts, T., Säde, M., Tõnisson, K. jt. Programmeerimine. Informaatika valikkursus gümnaasiumile. https://web.htk.tlu.ee/digitaru/programmeerimine/chapter/3-1-tsukkel/ (10.08.2023)

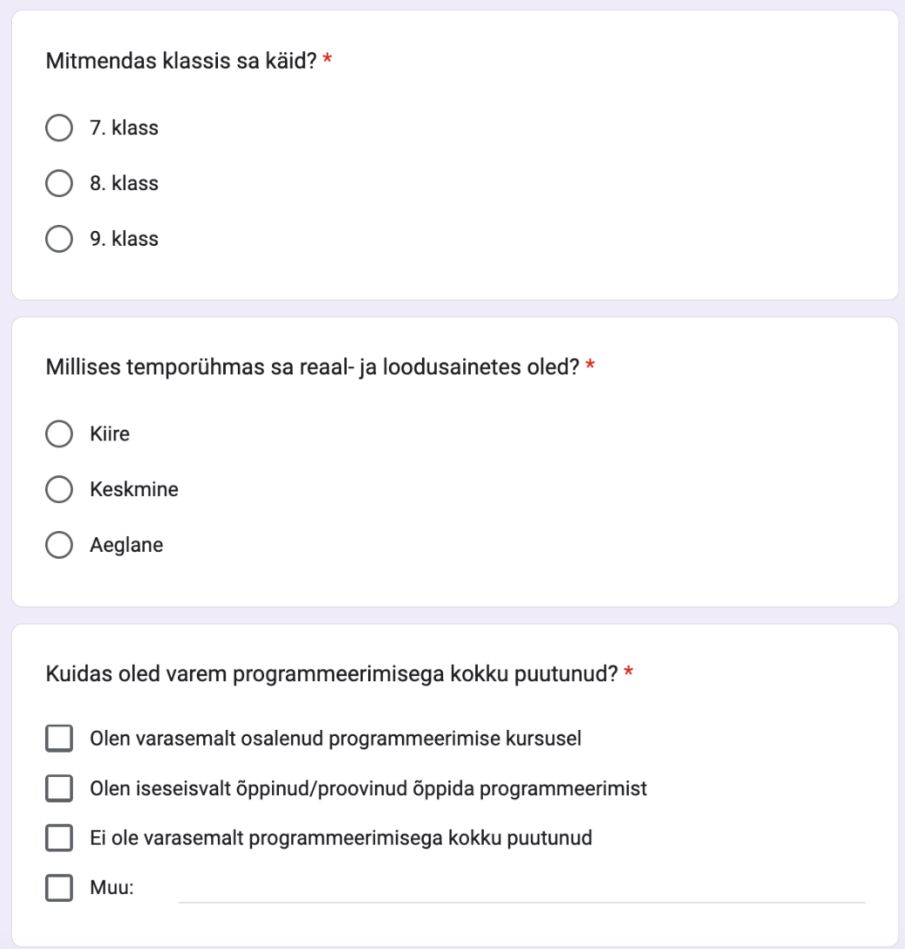
[27] Tartu Ülikooli arvutiteaduse instituudi informaatika didaktika töörühm. Programmeerimisest maalähedaselt. https://courses.cs.ut.ee/2023/progmaa/spring/Main/PARTVYlesanne

[28] *Collatz conjecture*. https://en.wikipedia.org/wiki/Collatz_conjecture

# Lisad

## I Kursuse alguses läbi viidud küsimustik

Piltide abil on esitatud kursuse alguses läbiviidud küsimustik uurimaks õpilaste tausta.



Milliste programmeerimiskeeltega oled varem kokku puutunud? *

- [ ] Python
- [ ] Java
- [ ] C++
- [ ] HTML
- [ ] CSS
- [ ] Scratch
- [ ] React/Vue
- [ ] Javascript
- [ ] Ei ole ühestki programmeerimis

Vasta, millises ulatuses nõustud järgmiste väidetega. *

| | 0 - Ei oska vastata. | 1 - Ei nõustu üldse. | 2 - Pigem ei nõustu. | 3 - Olen neutraalne. | 4 - Pigem nõustun. | 5 - Nõustun täielikult. |
|---|---|---|---|---|---|---|
| Kasutan igapäevaselt oma telefoni/tahvelarvutit/arvutit. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Mulle meeldib matemaatika. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Mulle meeldib lahendada mõistatusi ja ristsõnu. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Mulle meeldib mängida malet. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Olen huvitatud programmeerimise õppimisest. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Olen väga elevil programmeerimise õppimisest. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Arvan, et programmeerimine on raske. | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |

## II Praktikumide lõpus läbi viidud tagasiside ankeet

Pildi abil on esitatud iga praktikumi lõpus läbiviidud tagasiside ankeet.

Vasta, millises ulatuses nõustud järgmiste väidetega. *

| | 1 - Ei nõustu üldse. | 2 - Pigem ei nõustu. | 3 - Olen neutraalne. | 4 - Pigem nõustun. | 5 - Nõustun täielikult. |
|---|---|---|---|---|---|
| Mulle meeldis tänane praktikum. | ○ | ○ | ○ | ○ | ○ |
| Ülesanded olid minu jaoks jõukohased. | ○ | ○ | ○ | ○ | ○ |
| Jõudsin kõik ülesanded valmis. | ○ | ○ | ○ | ○ | ○ |
| Arvan, et programmeerimine on raske. | ○ | ○ | ○ | ○ | ○ |
| Ootan juba järgmist programmeerimise praktikumi. | ○ | ○ | ○ | ○ | ○ |

# III Ainekava

Piltide abil on esitatud loodud kursuse ainekava.

## Ainekava

### Programmeerimine III kooliastmele

**Õppeaine:** Programmeerimine III kooliastmele

**Klass:** 7.-9. Klass

**Maht:** pool kooliaastat

**Tundide arv:** 17 (90-minutilised)

**Vajalik taristu:** igale õpilasele arvuti, interneti võimalus, ekraani jagamise võimalus

#### Läbitavad teemad

- **Sissejuhatus programmeerimisse.** Programmeerimisega seotud tööpõhimõtted. Programm, protsess ja roll. Programmeerimiskeeled.
- **Algoritm.** Algoritmi mõiste, algoritmi koostamine ja realiseerimine. Etteantud tegevusjuhisest arusaamine, ise koostamine ja rakendamine. Andmete otstarbekalt muutmine. Lihtsamate tüüpalgoritmide kasutamine.
- **Andmed.** Objektid - nende omadused ja kasutamine.
- **Muutujad.** Muutuja kasutamine ja sellele väärtuse andmine.
- **Sisendid ja väljundid.** Sisendi andmine klaviatuurilt.
- **Tegevused ja avaldised.** Lihtsamad teksti-, loogika- ja arvavaldised.
- **Valikud.** Tingimuslaused *if* ja *else*.
- **Kordused.** Lõpmatu kordus, kordamine teatud arv kordi, kordamine etteantud tingimustel, kordus korduse sees.

#### Õpitulemused kursuse lõpetamisel

Kursust lõpetades õpilane

- mõistab ja kasutab teadlikult järgnevaid mõisteid: programm, protsess, algoritm, muutuja, avaldis, valik, tsükkel, funktsioon, programmeerimiskeel, sisend ja väljund;

- on suuteline analüüsima etteantud programmi töö tulemust ning selle käitumist enne käivitamist ennustada;
- oskab programmides teha otstarbekaid muudatusi ja/või täiendusi;
- oskab koostada programmi sõnalise kirjelduse alusel;
- oskab koostada lihtsamaid avaldisi ja algoritme;
- oskab olukorras, kus programm ei tee talle soovitud tööd, analüüsida koodi ja parandada vigu.

**Ainekava tabelina**

| Õppenädal | Tundide arv | Põhimõisted ja õppeteemad | Teoreetiline ja praktiline tegevus tunnis | Soovitavad õpitulemused ja hindamine | Lõiming | Kasutatav lisamaterjal |
|---|---|---|---|---|---|---|
| 1 | 1,5 | Algoritm, programm. | Tunni esimeses osas laseb õpetaja õpilastel teha avaküsitluse, et näha õpilaste üleüldist meelestatust programmeerimise õppimisel ning et saada ülevaade potentsiaalselt programmeerimist toetavatest faktoritest (muud huvialad, suhtumine reaal- ja loodusainetesse, jms). Seejärel selgitab õpetaja neile tunni teemat - selgitab, mis on programmeerimine, programm ja algoritm. Selgitatu peale täidavad õpilased töölehel esimese ülesande.<br>Tunni teises osas näitab õpetaja õpilastele, kuidas käivitada esimest programmi teise ülesande näitel, seejuures selgitades esimese programmi toimimist. Lisaks selgitab õpetaja õpilastele veel Turtle'i käske. Seejärel saavad õpilased ise katsetada (vajadusel õpetaja abiga) ise programmide loomist, proovides lahendada kolmandat ülesannet. Ülesande lahendamise abiks lisaks õpetajale on | Õpilane saab aru, mis on programmeerimine, programm ja algoritm, on suuteline koostama programmi, mis Turtle'i abil joonistab ettenähtud kujundeid, on võimeline lugema programmijuppi ja käivitamata saama aru selle toimimisest.<br><br>Hindamine on arvestuslik. | Õpilane võib Turtle'i abil koostada erinevaid joonistusi, mis on seotud reaalse eluga. Õpilane kasutab 7. klassi matemaatikas õpitavat hulknurga sisenurkade summa valemit efektiivselt. | Praktikum 1 töölehed |
| | | | lisaleht, kus on lahti selgitatud ülesande lahendamise loogika.<br>Enne tunni lõppu täidavad õpilased tundi kokkuvõtva küsimustiku, et saada aimu, kuidas õpilsed ise tunnevad, et neil tunnis läks. Lisaks on üks küsimus, mis annab õpetajale teada, kas tunniteema sai omandatud.<br><br>Ülesanded:<br>Selgita oma sõnadega, mis on algoritm.<br>Selgita oma sõnadega, mis on programm.<br>Esimese programmi käivitamine etteantud programmi alusel.<br>Kujundite joonistamine - ruut, võrdhaarne kolmnurk ja korrapärane kuusnurk. | | | |
| 2 | 1,5 | Andmetüübid, muutujad ja tehted, täisarv, ujukomaarv, sõne, tõeväärtus, andmetüüp, muutuja, | Tunni esimeses alustab õpetaja tunniteema seletamisega - seletab, mis on andmetüübid ja millised on erinevad andmetüübid. Seletatu peale lahendavad õpilased esimese ülesande. Peale seda annab õpetaja ülevaate sellest, kuidas saab Thonny abil kontrollida andmetüüpi ja kuidas seda vajadusel muuta, mille peale õpilased lahendavad teise ja kolmanda ülesande. | Õpilane saab aru, mis on andmetüüp ja millised on erinevad andmetüübid Pythonis, oskab arvuti abil kontrollida, mis andmetüübiga on tegu, | Õpilane seostab muutujanimede kirjutamist vastavalt sobivale keelele (eesti keel, inglise keel, vene keel, jne). Võimalik lahendada matemaatikaülesandeid ja seostada Pythoni | Praktikum 2 töölehed |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | väljund, ümardamine. | Tunni teises osas selgitab õpetaja õpilastele, mis on muutuja, kuidas muutujale väärtust saab anda ja millised nimed muutujaks sobivad või ei sobi. Selle põhjal lahendavad õpilased neljanda ja viienda ülesande.<br>Tunni kolmandas osas selgitab õpetaja õpilastele, milliseid tehteid teha saab ja mis on nende tehete prioriteetsuse järjekord. Selle alusel lahendavad õpilased kuuenda ülesande.<br>Enne tunni lõppu täidavad õpilased kokkuvõtva küsimustiku, et saada aimu, kuidas õpilased ise tunnevad, et neil tunnis läks. Lisaks on üks küsimus, mis annab õpetajale teada, kas tunniteema sai omandatud.<br>. | oskab teisendada andmetüüpe,<br>oskab ümardada ujukomaarve täisarvudeks vastavalt matemaatikas õpitud ümardamise reeglitele,<br>saab aru, mis on muutuja ja kuidas muutujale väärtust anda saab,<br>teab, milliste reeglite alusel muutujanimesid koostada võib,<br>teab, milliseid tehteid Pythonis teha saab,<br>teab, mis järjekorras Pythonis tehteid tehakse,<br>oskab ennustada, mis on Pythonis tehtava tehte tulemuseks.<br><br>Hindamine on arvestuslik. | andmetüüpe matemaatikas tuntud arvutüüpidega. Ümardamine kandub üle matemaatikast. | |
| 3 | 1,5 | Andmetüübid, muutujad, | Tunni esimeses osas kordab õpetaja eelmise tunni teemat - kordab üle, mis on | Õpilane | Õpilane seostab muutujanimede | Praktikum 3 töölehed |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | tehted, kasutaja sisend, täisarv, ujukomaarv, sõne, tõeväärtus, andmetüüp, muutuja, väljund, sisend, ümardamine. | andmetüübid ja millised on erinevad andmetüübid, mis on muutuja, kuidas muutujale väärtust saab anda ja millised nimed muutujaks sobivad või ei sobi.<br>Tunni teises osas selgitab õpetaja õpilastele, kuidas saada kasutajalt sisendit. Selle peale täidavad õpilased esimese ülesande. Peale seda selgitab õpetaja õpilastele, kuidas muuta saadud sisendi tüüpi, et seda vajadusel ekraanile väljastata, selgitades ülesande 1 allolevat näidet.<br>Tunni kolmandas osas lahendavad õpilased iseseisvalt ülesandeid. Lahendavad ära ülesanded 2, 3, 4, 5 ja 6.<br>Enne tunni lõppu täidavad õpilased kokkuvõtva küsimustiku, et saada aimu, kuidas õpilased ise tunnevad, et neil tunnis läks. Lisaks on üks küsimus, mis annab õpetajale teada, kas tunniteema sai omandatud. | saab aru, mis on andmetüüp ja millised on erinevad andmetüübid Pythonis,<br>oskab teisendada andmetüüpe,<br>oskab ümardada ujukomaarve täisarvudeks vastavalt matemaatikas õpitud ümardamise reeglitele,<br>saab aru, mis on muutuja ja kuidas muutujale väärtust anda saab,<br>teab, milliste reeglite alusel muutujanimesid koostada võib,<br>teab, milliseid tehteid Pythonis teha saab,<br>oskab ennustada, mis on Pythonis tehtava tehte tulemuseks.<br>saab aru, mis on kasutaja sisend, | kirjutamist vastavalt sobivale keelele (eesti keel, inglise keel, vene keel, jne). Lahendab matemaatikaülesandeid ja ka ümardamine kandub üle matemaatikast. | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | oskab küsida kasutajalt sisendit,<br>oskab kasutaja sisendiga teha operatsioone.<br><br>Hindamine on arvestuslik. | | |
| 4 | 1,5 | Tingimuslause. | Tunni esimeses osas kordab õpetaja eelmise tunni teemat - kordab üle, kuidas saada kasutajalt sisendit.<br>Tunni teises osas selgitab õpetaja õpilastele näidete abil, mis on tingimuslause. Selle peale täidavad õpilased esimese ülesande. Peale seda selgitab õpetaja õpilastele, kuidas tingimusi Pythonis kontrollida, ja räägib nendega läbi töölehtedel toodud näite parooli küsimise kohta. Koos vaadatakse läbi vastavale näitele sobiv plokkskeem ja lahendatakse teine ülesanne.<br>Tunni kolmandas osas lahendavad õpilased iseseisvalt ülesandeid. Lahendatakse ülesanded 3, 4, 5, 6, 7 ja 8.<br>Enne tunni lõppu täidavad õpilased kokkuvõtva küsimustiku, et saada aimu, kuidas õpilased ise tunnevad, et neil tunnis | Õpilane<br>saab aru, mis on tingimuslause,<br>oskab tingimuslausete alusel koostada plokkskeemi,<br>oskab praktiliselt rakendada tingimuslausete kasutamist.<br><br>Hindamine on arvestuslik. | Õpilane seostab tingimuslauseid igapäevaelu valikutega ja võrdlusi matemaatikas õpitud põhimõtetega. | Praktikum 4 töölehed |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | läks. Lisaks on üks küsimus, mis annab õpetajale teada, kas tunniteema sai omandatud. | | | |
| 5 | 1,5 | Tingimuslause, järjend, andmed. | Tunni esimeses osas kordab õpetaja eelmise tunni teemat - kordab üle tingimuslause põhimõtte näite abil.<br>Tunni teises osas selgitab õpetaja, kuidas tingimuslauseid efektiivselt koostada if, elif ja else abil. Lisaks selgitab, kuidas kasutada programmeerimiskeeles Python loogilis operaatoreid (and, or, not). Selle peale lahendatakse koos ülesanne 1.<br>Edasi selgitab õpetaja õpilastele näite abil järjendi olemust. Seejuures annab ülevaate järjenditega tehtavatest tegevustest (elementide arvu leidmine, järjendist minimaalse elemendi leidmine, jne)<br>Tunni kolmandas osas lahendatakse ülesanne 2, mille sooritamiseks on vaja aktiivselt kasutada kõiki tunnis õpitud teadmisi. Lisaks on üks küsimus, mis annab õpetajale teada, kas tunniteema sai omandatud. | Õpilane<br>oskab efektiivselt kirjutada selgeid tingimuslauseid,<br>kasutada tingimuslausete koostamisel loogilisi operaatoreid (and, or, not),<br>mõistab järjendi olemust,<br>oskab järjenditega tegevusi teha.<br><br>Hindamine on arvestuslik. | Õpilane seostab matemaatikas õpituga, kuidas kolme külje järgi teha kindlaks, kas kolmnurk eksisteerib. Samuti seostab igapäevaelu ja koolis muudes ainetes õpituga järjendi olemust. Inglise keelega seostab sõnu len, min, max, sum, sorted jne. | Praktikum 5 töölehed |
| 6 | 1,5 | Järjend, andmed. | Tunni esimeses osas kordab õpetaja eelmise tunni teemat - kordab üle tingimuslause | Õpilane<br>mõistab järjendi olemust, | Õpilane seostab igapäevaelu ja koolis | Praktikum 6 töölehed |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | põhimõtte näite abil ja selgitab üle järjendi olemuse. Kui eelmises tunnis jäi viimane ülesanne pooleli, siis tehakse ka see. Selle peale lahendavad õpilased ülesande 1. Edasi selgitab õpetaja järjendist elemendi leidmise indeksi abil ja ka vastupidise. Selle peale lahendavad õpilased ülesanne 2. Tundi lõpetab küsimus, mis annab õpetajale teada, kas tunniteema sai omandatud. | oskab järjenditega tegevusi teha,<br>saab aru, kuidas järjendist indeksi abil elementi kätte saada,<br>saab aru, kuidas elemendi alusel järjendist selle indeksit leida,<br><br>Hindamine on arvestuslik. | muudes ainetes õpituga järjendi olemust. Inglise keelega seostab sõnu len, min, max, sum, sorted jne. Samuti seostab indekseerimist matemaatikas õpituga. | |
| 7 | 1,5 | Tsükkel. | Tunni esimeses osas kordab õpetaja eelmise tunni teemat - selgitab üle järjendi ja selle indeksite olemuse.<br>Tunni teises osas selgitab õpetaja näite alusel tsükli olemust. Selle peale lahendatakse ülesanne 1. Seejärel annab õpetaja ülevaate, kuidas programmeerimiskeeles Python muutuja väärtust tsükli abil muuta. Seejärel lahendatakse ülesanded 2, 3, 4, 5 ja 6.<br>Tunni kolmandas osas lahendatakse iseseisvalt ülesaned 7, 8 ja 9. Nende lõpetamisel selgitab õpetaja, kuidas tsükli abil Turtle'is korrapäraseid kujundeid joonistada. Selle peale lahendavad õpilased | Õpilane<br>saab aru tsükli olemusest,<br>oskab tsüklit realiseerida,<br>saab aru, mis on lõpmatu kordus,<br>oskab kirjutada programmi, mis kordab teatud arv kordi,<br>oskab kirjutada programmi, mis kordab etteantud tingimusel.<br><br>Hindamine on arvestuslik. | Õpilane seostab tsükli olemust igapäevaelus esinevate kordustega. | Praktikum 7 töölehed |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | oskab kirjutada programmi, mis kordab etteantud tingimusel,<br>mõistab, kuidas realiseerida kordust korduse sees.<br><br>Hindamine on arvestuslik. | | |
| 12 | 1,5 | Projekt | Tunni esimeses osas selgitab õpetaja ära projekti olemuse. Seejuures annab teada, mis tingimustel tuleb neil enda kursust lõpetav projekt teha.<br>Tunni teises osas antakse õpilastele paber ja kirjutusvahend. Nende eesmärgiks on välja mõelda projekt, mida praktikumide 12-16 vältel teevad. Paberile tuleb kirja panna kõik vajalik informatsioon, mida suudavad enda projekti kohta ette mõelda. Seejärel alustavad projekti kirjutamisega.<br>Tunni kolmandas osas näitavad õpilased õpetajale oma tunnis tehtud progressi ja räägivad põgusalt, mida plaanivad järgmine tund teha. | Õpilane<br>saab aru projekti olemusest,<br>suudab enda oskuste baasil välja mõelda, milliseid projekte oleks realistlik teha,<br>oskab enda aega planeerida,<br>oskab rakendada õpitud teadmisi praktilisel kujul.<br><br>Hindamine on arvestuslik. | Õpilane seostab projekti igapäevaelu ja koolitööga. | Praktikumide 12-16 tööleht |
| 13 | 1,5 | | | | | |
| 14 | 1,5 | | | | | |
| 15 | 1,5 | | | | | |
| 16 | 1,5 | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | ülesanded 10 ja 11. Tundi lõpetab küsimus, mis annab õpetajale teada, kas tunniteema sai omandatud. | | | |
| 8<br>9 | 1,5<br>1,5 | Tsükkel | Tundide esimeses osas kordab õpetaja eelmise tunni teemat - selgitab üle tsükli olemuse näidete abil.<br>Tunni teises osas lahendavad õpilased iseseisvalt ülesanded 1, 2, 3 ja 4.<br>Tunni kolmandas osas küsib õpetaja küsimuse, mis annab õpetajale teada, kas tunniteema sai omandatud. | Õpilane<br>saab aru tsükli olemusest,<br>oskab tsüklit realiseerida,<br>saab aru, mis on lõpmatu kordus,<br>oskab kirjutada programmi, mis kordab teatud arv kordi,<br>oskab kirjutada programmi, mis kordab etteantud tingimusel.<br><br>Hindamine on arvestuslik. | Õpilane seostab tsükli olemust igapäevaelus esinevate kordustega. | Praktikumide 8 ja 9 tööleht |
| 10<br>11 | 1,5<br>1,5 | Tsükkel | Tundide esimeses osas kordab õpetaja eelmise tunni teemat - selgitab üle tsükli olemuse näidete abil.<br>Tunni teises osas lahendavad õpilased iseseisvalt ülesanded 1 ja 2.<br>Tunni kolmandas osas küsib õpetaja küsimuse, mis annab õpetajale teada, kas tunniteema sai omandatud. | Õpilane<br>saab aru tsükli olemusest,<br>oskab tsüklit realiseerida,<br>saab aru, mis on lõpmatu kordus,<br>oskab kirjutada programmi, mis kordab teatud arv kordi, | Õpilane seostab tsükli olemust igapäevaelus esinevate kordustega. | Praktikumide 10 ja 11 tööleht |

| | | | | | | |
|---|---|---|---|---|---|---|
| 17 | 1,5 | Algoritm, programm, andmetüübid, muutujad ja tehted, täisarv, ujukomaarv, sõne, tõeväärtus, andmetüüp, muutuja, väljund, ümardamine, tingimuslause, järjend, andmed, tsükkel. | Tunni esimeses osas (30 min) sooritavad õpilased testi seni õpitu peale. Test toimub suletud materjalidega iseseisvalt.<br>Tunni teises osas (60 min) lahendavad õpilased etteantud juhise järgi kursuse teemasid hõlmava programmeerimisülesande. Lahendamine toimub iseseisvalt, kuid selle vältel võib kasutada enda poolt varem tehtud programme.<br>Tunni lõpus tehakse õpilastele kursust lõpetav küsimustik. | Õpilane<br>on suuteline ennustama programmi tööd,<br>oskab etteantud juhise järgi rakendada kursusel õpitud teadmisi.<br><br>Töid ei hinnata. | | Tööleht, kus on kirjas teooria töö ülesanded.<br>Tööleht, kus on kirjas programmi koostamise ülesanne. |

## IV Praktikumi 1 töölehed

Piltide abil on esitatud praktikumi 1 töölehtede sisu.

# PRAKTIKUM 1

## ALGORITM, PLOKKSKEEM, PROGRAMM, KILPKONNAGRAAFIKA

**Algoritm** – astmeline tegevusjuhis mingi tegevuse sooritamiseks.


- Algoritmi iga samm peab olema täpne ehk üheselt mõistetav.
- Algoritm peab olema lõplik ehk lõpliku arvu sammude järel viima lõpptulemusele.
- Algoritm peab olema efektiivne ehk andma probleemile korrektse vastuse.
- Algoritm peab olema üldine ehk lahendama ülesande iga eksemplari.

**Plokkskeem** – graafiline notatsioon algoritmide esitamiseks.

Plokkskeemis kasutatavad kujundid:

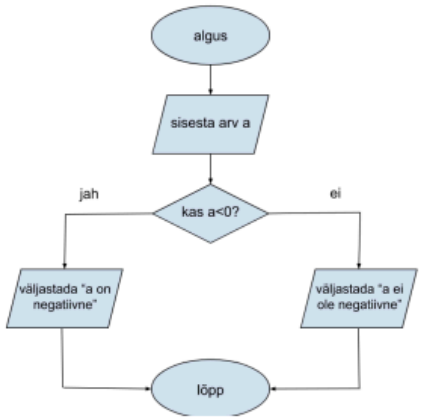
- ovaal – algus või lõpp,

- ristkülik – protsess, tegevused,

- rööpkülik – sisend või väljund,

- romb – otsustus,

- nool – tegevuste järgnevuse suund,




*Joonis 1. Plokkskeem kontrollimaks, kas arv on negatiivne.*

### ÜLESANNE 1

Joonisel 1 on välja toodud plokkskeem kontrollimaks, kas sisestatud arv on negatiivne. Tee antud plokkskeem ümber selliselt, et väljastataks õigesti järgmised väited: „a on null“, „a on positiivne“ ja „a on negatiivne“.

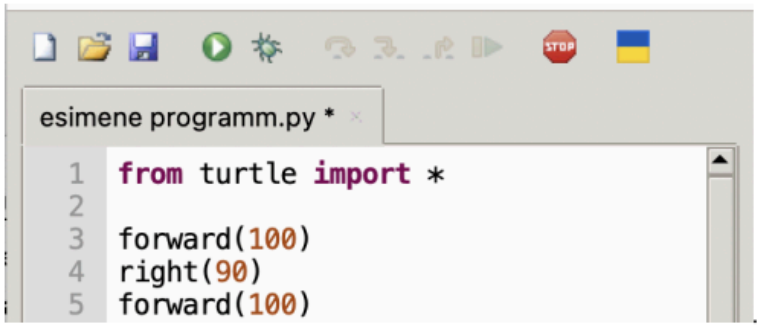
**Programm** – arvutile arusaadav algoritm, mis on kirjutatud kindlas programmeerimiskeeles.

## ÜLESANNE 2

Selleks, et näha programmi töötamist visuaalselt, kasutame kilpkonnagraafikat ehk Pythoni moodulit Turtle.

**Pane tähele!** Selleks, et kasutada funktsioone, mis asuvad Turtle moodulis, tuleb programmis esimesele reale alati kirjutada `from turtle import *`.

Kirjuta esimene testprogramm ja käivita see, vajutades rohelist noolega nuppu või klaviatuuril F5.




Kui käivitad programmi esimest korda, küsitakse siis arvuti küsib, kuhu programm salvestada – vali endale sobiv kaust, anna failile nimi ja salvesta.

**Pane tähele!** Faili nimeks ei tohi olla *turtle*.

## TURTLE'I KÄSUD

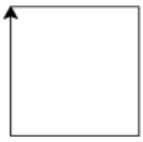
| | |
|---|---|
| `forward(x)` | Nool liigub edasi x pikslit. |
| `backward(x)` | Nool liigub tagasi x pikslit. |
| `right(x)` | Nool pöörab x kraadi paremale. |
| `left(x)` | Nool pöörab x kraadi vasakule. |
| `up()` | Nool enam ei joonista (sarnane päris elus pliiatsiotsa paberilt üles tõstmisega) |
| `down()` | Nool hakkab jälle joonistada (sarnane päris elus pliiatsiotsa paberile panemisega). |
| `speed(x)` | Muudab joonistamise kiirust. |
| `color("värv")` | Muudab pliiatsi värvi. **Pane tähele!** Värv tuleb kirjutada inglise keeles. |
| `begin_fill()`<br>`end_fill()` | Kujundite seest värvimiseks tuleb enne joonistamise alustamist kirjutada käsk `begin_fill()` ja peale joonistamise lõpetamist kirjutada käsk `end_fill()`. |

**Pane tähele!** Kui kirjutad rea ette trellid (#), siis seda osa peetakse koodis kommentaariks ja trellide järel olevat koodi ei sooritata.

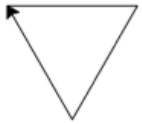
## ÜLESANNE 3

Joonista Turtle'i abiga järgmised kujundid. Kasuta ülesande kõrval olevat vaba ruumi vajalike arvutuste tegemiseks.

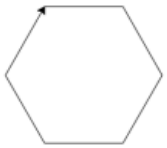
1. Korrapärane nelinurk



2. Korrapärane kolmnurk



3. Korrapärane kuusnurk



**Pane tähele!** Kui jääd hätta, siis järgmisel lehel on vihjed ja näide, kuidas nendele ülesannetele läheneda.

## ÜLESANNE 4

Joonista Turtle'i abiga pilt. Pildi sisu võid ise valida. Proovi kasutada ka erinevaid pliiatsi värve ja värvida kujundeid seest. Kasuta siin all olevat ruumi selleks, et panna mõtted paberile, mida joonistada plaanid.

## VIHJED ÜLESANDE 3 LAHENDAMISEKS

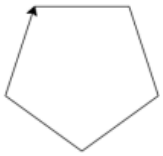
Korrapärase hulkhulga sisenurkade summa saad arvutada valmiga $(n-2)\cdot 180°$, kus $n$ tähistab nurkade arvu.

Et leida, mitu kraadi peaks üks nurk kujundil olema, saad jagada sisenurkade summa (valemiga leitud) nurkade arvuga.

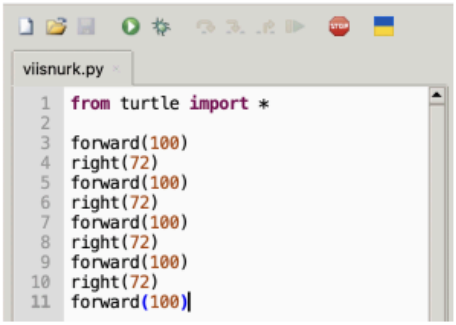
Selleks, et pöörata, pead hoopis arvestama kujundi välisnurkadega. Kuna pliiatsi käsk `forward(n)` liigub sirgjooneliselt, siis välisnurga saad arvutada, lahutades 180 kraadist sisenurga väärtuse.

## NÄIDE ÜLESANDE 3 LAHENDAMISEKS

Tahaksin joonistada korrapärast viisnurka.



- Pean arvutama, mis on selle kujundi sisenurkade summa.
  $(5-2)\cdot 180° = 3\cdot 180° = 540°$
- Et viisnurgal on viis nurka, siis saamaks teada, mitu kraadi on üks sisenurk, pean sisenurkade summa jagama viiega.
  $540° \div 5 = 108°$
- Et pean kujundi joonistamisel arvestama välisnurkadega, siis pean arvutama nurga välisnurga.
  $180° - 108° = 72°$
- Selleks, et nüüd joonistada korrapärast viisnurka, peab mu programm välja nägema järgmine.



## V Praktikumi 2 töölehed

Piltide abil on esitatud praktikumi 2 töölehtede sisu.

# PRAKTIKUM 2

## ANDMETÜÜBID, MUUTUJAD, TEHTED

- `int` – andmetüüp täisarvude jaoks (näiteks `5`, `-10`, `0`).
- `float` – andmetüüp koma sisaldavatele arvudele (näiteks `4.5`, `-7.23`, `10.0`).

  **Pane tähele!** Pythonis kasutatakse koma asemel punkti.
- `str` – andmetüüp tekstide jaoks (näiteks `"programmeerimine"`, `'programmeerimine'` või `"programmeerimine on tore"`).
- `bool` – andmetüüp tõeväärtuse jaoks (näiteks `True` ja `False`).

  **Pane tähele!** Tõeväärtused kirjutame Pythonis suure algustähega.

Et saada teada, mis andmetüübiga on tegu, võime Thonnys selle ekraanile väljastada. Selleks kasutame käsku `print`. Käsk `print` kuvab sulgude sees info ekraanile.

```
>>> print(type("programmeerimine"))
 <class 'str'>
```

Andmetüüpide vahel saab teha ka teisendusi.

```
>>> print(type("2"))
 <class 'str'>
>>> print(type(int("2")))
 <class 'int'>
```

Olgu meil sõne `"2"`. Nagu vasakul näha, siis tegemist on `str` andmetüübiga. Küll aga ei saa teha sõnega tehteid – me ei saa liita sõnele `"2"` juurde `3`, et saada `5`, nagu me ei saa sõnele `"koer"` liita `3`. Et saaksime sõnega `"2"` teha tehteid, peame selle enne teisendama täisarvuks selliselt, nagu vasakul näidatud. Teisendusi saab teha ka teiste andmetüüpide vahel.

### ÜLESANNE 1

1. Ennusta, mis väljastatakse ekraanile, kui kirjutada `print(int(5.7))`.

_____

2. Selgita, miks sa eelmises alapunktis just selle arvu pakkusid.

_____

3. Kontrolli nüüd Thonny abil, mis väljastatakse ekraanile, kui kirjutada `print(int(5.7))`.

_____

4. Selgita, miks eelmises alapunktis just selline arv väljastati.

_____

5. Kirjuta funktsiooni väljakutse, millega saaksime `5.7` teisendada täisarvuks ümardamisreeglite järgi õigesti.

_____

## ÜLESANNE 2

Vali igale küsimusele õige valikvastus ja kirjuta kõrval olevale vabale alale põhjendus, miks just see vastusevariant õige on.

1. Mis ilmub ekraanile, kui kirjutada `print(type("100"))`?

   a. `<class 'int'>`

   b. `<class 'float'>`

   c. `<class 'str'>`

   d. Veateade.

2. Mis ilmub ekraanile, kui kirjutada `print(type(100,0))`?

   a. `<class 'int'>`

   b. `<class 'float'>`

   c. `<class 'str'>`

   d. Veateade.

3. Mis ilmub ekraanile, kui kirjutada `print(type(tere))`?

   a. `<class 'int'>`

   b. `<class 'float'>`

   c. `<class 'str'>`

   d. Veateade.

4. Mis ilmub ekraanile, kui kirjutada `print(type(false))`?

   a. `<class 'int'>`

   b. `<class 'float'>`

   c. `<class 'str'>`

   d. Veateade.

5. Mis ilmub ekraanile, kui kirjutada `print(type(100))`?

   a. `<class 'int'>`

   b. `<class 'float'>`

   c. `<class 'str'>`

   d. Veateade.

**Muutuja** – kindel mälupiirkond, kus on mingi kindel väärtus. Muutujal on nimi ja väärtuse andmine käib võrdusmärgi abil.

```
1 nimi = "minu nimi"
2 vanus = 20
3 kas_täisealine = True
4 pikkus = 170.5
```

**Pane tähele!** Muutuja nime valimisel tuleb arvestada järgmiste reeglitega:

1. Nimi **ei tohi** sisaldada tühikut ega erimärke peale alakriipsu. Tühiku asemel saab kasutada alakriipsu või kirjutada sõnad kokku ilma tühikuteta.
   *Näiteks ei kirjuta me muutujaks ema nimi, vaid ema_nimi või emaNimi.*
2. Muutuja nime esimene sümbol **ei tohi** olla number ja **hea tava** on alustada väikese tähega - peale esimese tähe võib olla numbreid.
   *Näiteks ei sobi muutujateks 7pöialpoissi, aga sobib seitsePö1alpoissi või seitse_pö1alpoissi.*
3. Nimi **ei tohi** olla Pythonis eritähendusena.
   *Näiteks ei tohi muutuja nimena kasutada järgmisi sõnu: and, def, elif, else, for, from, if, import, not, in, or, pass, return, while, True, False, None, jne.*

## ÜLESANNE 3

Kirjuta iga nime juurde, kas see sobib muutuja nimeks või mitte. Kui ei sobi, siis põhjenda, mis reeglit see rikub. Kui muutuja nimi sobib, aga tegemist on halva tavaga, siis too see ka välja.

1. `AbcDef_123`

---

2. `AbcDef-123`

---

3. `1_abc`

---

4. `_abc`

---

5. `return`

---

Tehete tegemisel tuleb appi matemaatikast õpitud tehete tähtsuse järjekord. Ainus erisus tekib sellest, et tavalises matemaatikas ei ole meil otseselt jäägi leidmiseks eraldi tehet.

1. astendamine,
2. korrutamine ja jagamine,
3. jäägi leidmine,
4. liitmine ja lahutamine

**Pane tähele!** Samamoodi nagu tavalises matemaatikas saab ka Pythonis näidata tehete prioriteete sulgudega.

**TEHTED PYTHONIS**

| Märk | Näide | Selgitus |
|---|---|---|
| + | `3 + 2 = 5` | Liitmine |
| - | `3 - 2 = 1` | Lahutamine |
| * | `3 * 2 = 6` | Korrutamine |
| / | `3 / 2 = 1.5` | Jagamine |
| // | `7 // 3 = 2` | Täisarvuline jagamine, ümardab alla |
| % | `7 % 4 = 3` | Jäägi leidmine |
| ** | `3 ** 2 = 9` | Astendamine |

**ÜLESANNE 4**

Vasta, mis väljastatakse järgmiste tehete tulemusena ekraanile. Arvutamiseks võid kasutada iga alaülesande all antud vaba ruumi.

1. `print(-12 / (-4) / 8 - (-6))`

_____

2. `print(int(-12 / (-4) / 8 - (-6)))`

_____

3. `print(-12 % (-4) / 8 - (-6))`

_____

4. `print(-13 // (-4) / 8 - (-6))`

_____

## ÜLESANNE 5

Mis andmetüübi me saame, kui jagame ühest täisarvust teise?

_____

## ÜLESANNE 6

Kirjuta iga tehte juurde, kas see annab vastuseks 2. Põhjenda igat vastust.

1. `7 // 3`

_____

2. `-11 // (-5)`

_____

3. `5.0 // 2`

_____

4. `7 % 5`

_____

5. `12 / 6`

_____

## * ÜLESANNE 7

Joonista Turtle'i abiga mingi pilt. Seejuures sel korral salvesta väärtused muutujatesse. Ürita leida palju mustreid, et saaksid ühte muutujat rakendada mitmes erinevas kohas.

## VI Praktikumi 3 töölehed

Piltide abil on esitatud praktikumi 3 töölehtede sisu.

# PRAKTIKUM 3

## ANDMETÜÜBID, KASUTAJA SISEND

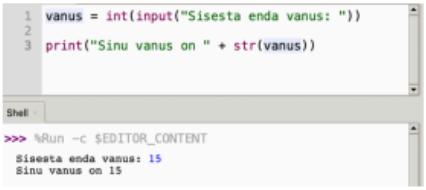
Selleks, et programm saaks kasutajalt küsida sisendit, tuleb kirjutada `input("Sisesta midagi: ")`. See laseb kasutajal klaviatuuril midagi sisestada. Vaikimisi on sisestatud väärtuse andmetüüp sõne ehk `str`.

**Pane tähele!** Teksti sulgude sees saab muuta.

Küll aga vahel soovime piirata seda, milliseid andmeid kasutaja meile anda saab. Näiteks, kui küsime kasutajalt vanust, ei sobi anda vastuseks „kass“. Seega peaks piirama, et soovime temalt saada ainult täisarve.

- Kui tahame, et kasutaja sisestaks meile täisarvu, siis tuleb kirjutada `int(input("Sisesta midagi: "))`.
- Kui tahame, et kasutaja sisestaks meile ujukomaarvu, siis tuleb kirjutada `float(input("Sisesta midagi: "))`.

**Pane tähele!** Nüüd saab kasutaja sisestada ainult kindlaid andmetüüpe selliselt, et programm jätkaks tegevust. Mingi muu andmetüübiga väärtuse lisamisel annab programm veateate ja lõpetab tegevuse.




**Pane tähele!** Kui soovime koos täpsustava tekstiga ekraanile kuvada mingit arvulist väärtust, peame selle enne väljastamist teisendama sõneks. Seda seepärast, et lausetele saab otsa liita ainult teisi lauseid.

### ÜLESANNE 1

Kirjuta programm, mis küsib kasutajalt sisendiks lause ja väljastab selle ekraanile.

```
>>> %Run lause.py
  Sisesta lause: Programmeerimine on tore
  Programmeerimine on tore
```

Õigesti kirjutatud programm peaks toimima selliselt, nagu kõrval välja toodud on.

### ÜLESANNE 2

Kirjuta programm, mis küsib kasutajalt sisendiks kaks täisarvu `a` ja `b`.

1. Salvesta muutujasse `summa` nende kahe muutuja summa. Ekraanile tuleks väljastada `summa` väärtus.
2. Salvesta muutujasse `vahe` nende kahe muutuja vahe. Ekraanile tuleks väljastada `vahe` väärtus.
3. Salvesta muutujasse `jagatis` nende kahe muutuja jagatis. Ekraanile tuleks väljastada `jagatis` väärtus.

4. Salvesta muutujasse `jääk` nende kahe muutuja jagamise tulemusel saadud jääk. Ekraanile tuleks väljastada `jääk` väärtus.

Õigesti kirjutatud programm peaks toimima selliselt, nagu kõrval välja toodud on.

```
>>> %Run tehted.py

  Sisesta esimene täisarv: 100
  Sisesta teine täisarv: 6
  Summa: 106
  Vahe: 94
  Jagatis: 16.666666666666668
  Jääk: 4
```

**Pane tähele!** Täisarvude jagamisel tavaline jagamine annab vastuseks ujukomaarvu.

## ÜLESANNE 3

Kirjuta programm, mis lahendab järgmise ülesande:

*Isikul 1 on x kommi. Isikul 2 on y kommi rohkem. Mitu kommi on isikutel kokku?*

Kommide koguse peaks programm küsima kasutajalt.

Õigesti kirjutatud programm peaks toimima selliselt, nagu kõrval välja toodud on.

```
>>> %Run kommid.py

  Mitu kommi on isikul 1? 56
  Mitu kommi rohkem on isikul 2? 37
  Isikutel on kokku 149 kommi
```

## ÜLESANNE 4

Kirjuta programm, mis küsib kasutajalt ühe täisarvu. Seejärel liidab sellele 4, korrutab tulemuse 2ga, lahutab 8, jagab esialgse arvuga ning lõpuks liidab seitse. Olgu iga uus väärtus salvestatud uude muutujasse.

Katseta programmi erinevate arvudega. Mida märkad?

---

## ÜLESANNE 5

Kirjuta programm, mis küsib kasutajalt ristküliku pikkuse ja laiuse täisarvudena. Seejärel arvutab ristküliku ümbermõõdu ja pindala ning väljastab need ekraanile.

Õigesti kirjutatud programm peaks toimima selliselt, nagu kõrval välja toodud on.

```
>>> %Run 'ristkülik.py'

  Sisesta ristküliku pikkus: 18
  Sisesta ristküliku laius: 14
  Ristküliku ümbermõõt on 64
  Ristküliku pindala on 252
```

## ÜLESANNE 6

Kirjuta programm, mis küsib kasutajalt sisendiks ühe ujukomaarvu. Seejärel muutujasse `ümardatud_arv` ümardab selle arvu kahe komakohani.

Õigesti kirjutatud programm peaks toimima selliselt, nagu kõrval välja toodud on.

```
>>> %Run 'ümardamine.py'

  Sisesta ujukomaarv: 3.14159265359
  3.14
```

## * ÜLESANNE 7

Kirjuta programm, mis küsib kasutajalt täisnurkse kolmnurga kaks kaatetit ning arvutab selle hüpotenuusi. Küljed peaksid olema sisestatavad ka ujukomaarvudena.

Õigesti kirjutatud programm peaks toimima selliselt, nagu all välja toodud on.

```
>>> %Run 'hüpotenuus.py'

  Sisesta täisnurkse kolmnurga esimene kaatet: 15
  Sisesta täisnurkse kolmnurga teine kaatet: 20
  Täisnurkse kolmnurga hüpotenuus on: 25.0
```

**Pane tähele!** Kaatetite ja hüpotenuusi vahel eksisteerib seos $c^2 = a^2 + b^2$.

**Pane tähele!** Ruutjuurt saab võtta selliselt, et esimesele reale kirjutad `import math` ja ruutjuure saamiseks kasutad funktsiooni `math.sqrt()`.

## * ÜLESANNE 8

Kirjuta programm, mis küsib kasutajalt silindri kõrguse ja põhja raadiuse. Seejärel programm väljastab silindri täispindala ja ruumala. Ümarda vastused kahe komakohani.

```
>>> %Run silinder.py

  Sisesta silindri kõrgus: 27
  Sisesta silindri põhja raadius: 5
  Silindri ruumala on 2120.58
  Silindri täispindala on 1005.31
```

Õigesti kirjutatud programm peaks toimima selliselt, nagu kõrval välja toodud on.

**Pane tähele!** Arvu $\pi$ saab, kirjutades esimesele reale `import math` ja kasutades `math.pi`.

Kasuta all olevat ruumi, et teha vajalikke arvutusi ja märkmeid.

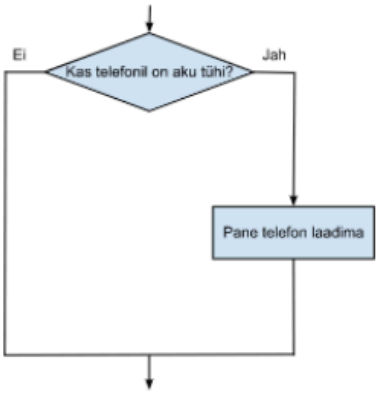
## VII Praktikumi 4 töölehed

Piltide abil on esitatud praktikumi 4 töölehtede sisu.

# PRAKTIKUM 4

## TINGIMUSLAUSE

Tihti on vaja teha midagi vaid mingil kindlal tingimusel.




*Näiteks „Kui telefonil on aku tühi, siis pane see laadima" korral on kirjeldatud, mida tuleb teha, kui tingimus on täidetud.*
*Tingimus: „Kui telefonil on aku tühi".*
*Tegevus: „Pane telefon laadima".*
*Tegevus, kui tingimus ei ole täidetud: puudub.*

**Pane tähele!** Kui tingimus ei ole täidetud, siis see samm lihtsalt jäetakse vahele.

Tihti on määratud tegevus ka siis, kui tingimus ei kehti. Meie näite puhul oleksime saanud öelda näiteks, et kui telefonil ei ole aku tühi, siis me ei jätka selle kasutamist.
Selliseid valikuid nimetatakse **tingimuslauseteks**.

### ÜLESANNE 1

Joonista plokkskeem järgmiste tingimuste kohta:

- kui sul on raamatukogu raamat alles, siis tagasta see,
- kui raamatukogu raamat ei ole alles, siis osta raamatukogule uus raamat.

## TINGIMUSTE KONTROLLIMINE PYTHONIS

| Märk | Näide | Näite selgitus |
|---|---|---|
| < | a < 4 | a on väiksem kui 4 |
| > | a > 4 | a on suurem kui 4 |
| <= | a <= 4 | a on väiksem kui 4 või sellega võrdne. |
| >= | a >= 4 | a on suurem kui 4 või sellega võrdne |
| == | a == 4 | a on võrdne arvuga 4 |
| != | a != 4 | a ei ole võrdne arvuga 4 |

Näiteks, kui tahaksime kirjutada programmi, mis küsib kasutajalt tema parooli ja kontrollib, kas see on õige, siis programm võiks

- õige parooli sisestamisel öelda „Õige parool!“,
- vale parooli sisestamisel öelda „Vale parool!“.

Selleks peaksime kasutajalt esimese asjana küsima tema parooli. Eelmises praktikumis õppisime, et seda saab teha `input()` abiga.

```
1 parool = str(input("Sisesta parool: "))
```

Nüüd oleks tarvis kontrollida, kas kasutaja sisestatud parool on õige.

Selleks kasutamegi tingimuslauset.

**Tingimuslauses** on tingimus, mille alusel otsustatakse, kuidas programm edasi toimib.

```
1 parool = str(input("Sisesta parool: "))
2
3 if parool == "minuparool123":
4     print("Õige parool!")
```

**Tingimus** on tõeväärtuse tüüpi (`boolean`) avaldis, mis on kas tõene (`True`) või väär (`False`). `True` on see siis, kui tingimus on täidetud ja `False` siis, kui tingimus ei ole täidetud.

**Pane tähele!** Ära unusta `if`-rea lõpust koolonit (`:`).

**Pane tähele!** `print` on taandatud võrreldes if-reaga. Selleks tuleb kasutada TAB klahvi või vajutada tühikut neli kohta. Selliselt toimimine määrab ära, et `print` käsk täidetakse `if`-tingimusest sõltuvalt. Seda nimetatakse **treppimiseks**.

Meie programm töötab, aga ainult juhul, kui kasutaja sisestatud parool oli õige. Kui sisestatud parool on vale, siis programm lõpetab töö midagi edasi tegemata.

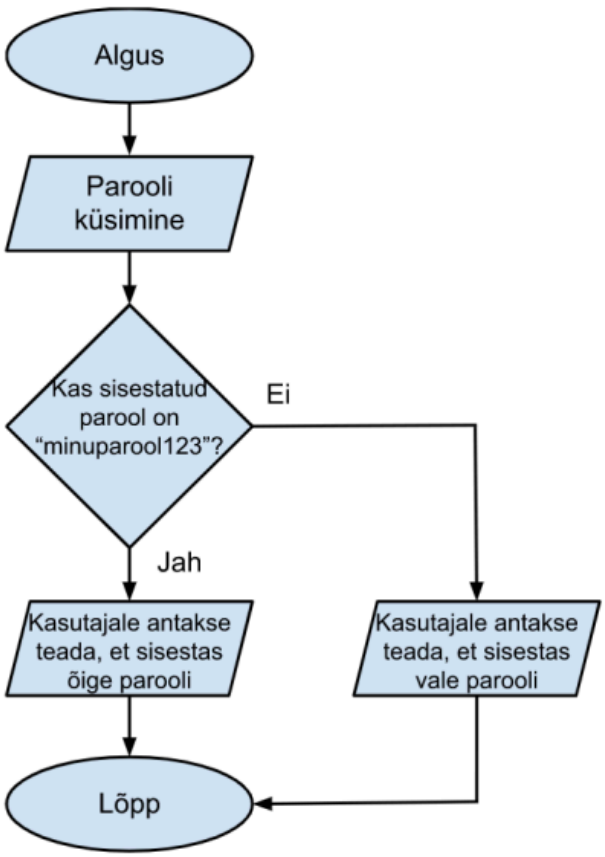
Selleks, et programm annaks meile teada, kui sisestatud parool on vale, tuleb kirjutada programmi `else`-osa, mis juhtub siis kui tingimuslause ei ole täidetud ehk selle tõeväärtus on `False`.

raamatukogu_raamat.py

```
1 parool = str(input("Sisesta parool: "))
2
3 if parool == "minuparool123":
4     print("Õige parool!")
5 else:
6     print("Vale parool!")
```

Kirjuta see programm endale arvutisse ümber ja katseta selle töötamist. Proovi vahetada ka soovitud parooli.

Selle programmi plokkskeem näeks välja järgmine:




## ÜLESANNE 2

```
1  sõne = "kiisu" + "Miisu"
2
3  if sõne == "kiisumiisu":
4      print("Sõned on samad")
5  else:
6      print("Sõned ei ole samad")
```

Mis väljastatakse ekraanile?

a. kiisumiisu

b. kiisuMiisu

c. Sõned on samad

d. Sõned ei ole samad

e. Veateade

## ÜLESANNE 3

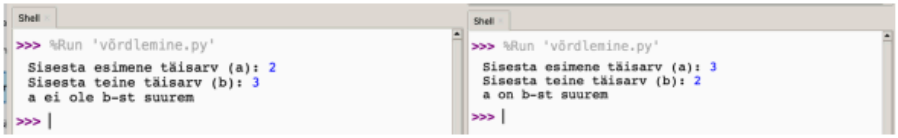
Kirjuta programm, mis küsib kasutajalt ühe täisarvu. Seejärel, kui täisarv on naturaalarv (suurem kui 0 või võrdne sellega), siis väljastatakse ekraanile "Arv on naturaalarv". Kui täisarv on negatiivne (väiksem kui 0), siis väljastatakse ekraanile "Arv ei ole naturaalarv".

Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:


## ÜLESANNE 4

Kirjuta programm, mis küsib kasutajalt kaks täisarvu a ja b. Kui a on b-st suurem, siis ekraanile väljastakse "a on b-st suurem". Kui a ei ole b-st suurem, siis ekraanile väljastatakse "a ei ole b-st suurem".

Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:




**Pane tähele!** Kui a ja b on võrdsed, siis ekraanile väljastatakse ikka "a ei ole b-st suurem", sest hetkel me ei kontrolli olukorda, kus a ja b on omavahel võrdsed.

## ÜLESANNE 5

Kirjuta programm, mis küsib kasutajalt, kui vana ta on. Seejärel väljastab ekraanile, kas ta on täisealine või mitte.

Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:


## ÜLESANNE 6

Kirjuta programm, mis küsib kasutajalt kaks sõne. Seejärel kontrollib, kas sisestatud sõned on samad või mitte.

Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:


## ÜLESANNE 7

Kirjuta programm, mis küsib kasutajalt pitsa läbimõõdu (diameetri), arvutab selle pitsa pindala ja seejärel väljastab ekraanile, kas tegemist on suure, keskmise või väikse pitsaga.

Pitsa on väike, kui selle pindala on kuni 500 $cm^2$ või võrdne sellega.

Pitsa on keskmine, kui selle pindala on on suurem kui 500 $cm^2$ ja väiksem kui 1300 $cm^2$.

Pitsa on suur, kui selle pindala on on suurem kui 1300 $cm^2$ või võrdne sellega.

Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:

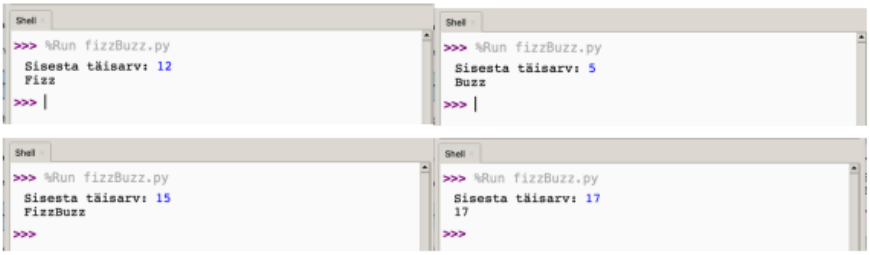

**Pane tähele!** Uuri internetist *math.pi* konstanti.

## * ÜLESANNE 8

Kirjuta programm, mis küsib kasutajalt ühte täisarvu.

1. Kui täisarv jagub 3-ga, siis trükib ekraanile sõna Fizz.
2. Kui täisarv jagub 5-ga, siis trükib ekraanile sõna Buzz.
3. Kui täisarv jagub nii 3-ga kui ka 5-ga, siis trükib programm ekraanile sõna FizzBuzz.
4. Muudel juhtudel trükib ekraanile sisestatud arvu.

Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:




Uuri internetist järgmisi märksõnu:

- Python `elif`
- Python operators (`and`)

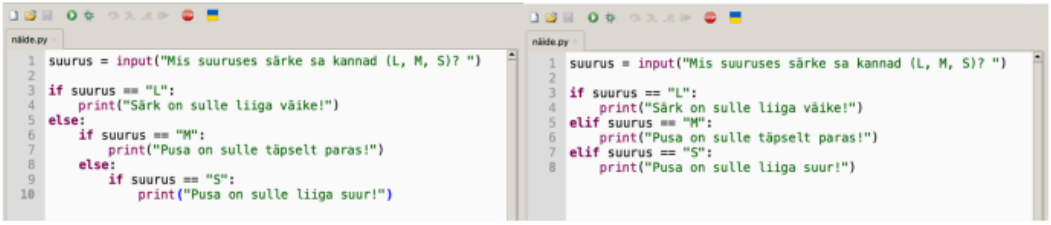
## VIII Praktikumi 5 töölehed

Piltide abil on esitatud praktikumi 5 töölehtede sisu.

# PRAKTIKUM 5

## TINGIMUSLAUSE JA JÄRJEND

Vahepeal võib tulla ette olukordi, kus on vaja kontrollida mitut tingimust. Selle jaoks ei ole mõttekas kirjutada mitu erinevat `if`-lauset.



```
näide.py
1  suurus = input("Mis suuruses särke sa kannad (L, M, S)? ")
2
3  if suurus == "L":
4      print("Särk on sulle liiga väike!")
5  else:
6      if suurus == "M":
7          print("Pusa on sulle täpselt paras!")
8      else:
9          if suurus == "S":
10             print("Pusa on sulle liiga suur!")
```

```
näide.py
1  suurus = input("Mis suuruses särke sa kannad (L, M, S)? ")
2
3  if suurus == "L":
4      print("Särk on sulle liiga väike!")
5  elif suurus == "M":
6      print("Pusa on sulle täpselt paras!")
7  elif suurus == "S":
8      print("Pusa on sulle liiga suur!")
```

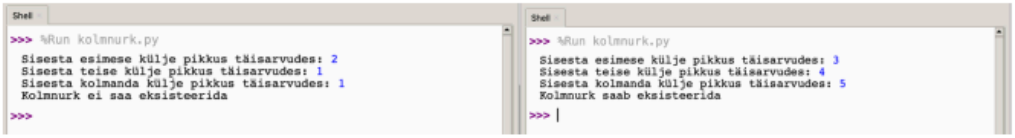
Näiteks on programm vasakul üsna segane. Kui neid tingimusi oleks veel, oleks kood veelgi segasem. Pythonis on kasutusel `elif`, mis on kombinatsioon `else`'ist ja `if`ist.

Vahepeal on vaja kontrollida mitut erinevat tingimust. Selliseid tingimusi saab esitada loogiliste tehete abil.

| | | |
|---|---|---|
| **and** | `x > 5 and x < 10`<br>Kui x on 6, siis tagastatakse *True*.<br>Kui x on 4, siis tagastatakse *False*.<br>Kui x on 11, siis tagastatakse *False*. | Tagastatakse `True`, kui mõlemad tingimused on tõesed |
| **or** | `x > 5 or x < 10`<br>Kui x on 6, siis tagastatakse *True*.<br>Kui x on 4, siis tagastatakse *True*.<br>Kui x on 11, siis tagastatakse *True*. | Tagastatakse `True`, kui vähemalt üks tingimus on tõene |
| **not** | `not(x > 5 and x < 10)`<br>Kui x on 6, siis tagastatakse *False*.<br>Kui x on 4, siis tagastatakse *True*.<br>Kui x on 11, siis tagastatakse *True*. | Tagastatakse sulgude sees oleva tulemuse vastand. Kui sulgude sees tingimuslause väärtus oleks `True`, siis tagastatakse `False`. |

## ÜLESANNE 1

Kirjuta proramm, mis küsib kasutajalt alguses kolmnurga ühe külje, siis teise külje ja lõpuks kolmanda külje.
Seejärel programm peab kasutajale väljastama, kas selline kolmnurk saab eksisteerida.
Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:




**Pane tähele!** Kahe külje summa peab olema alati suurem kui kolmas külg.

Seni oleme me defineerinud ära iga muutuja eraldi. Reaalsuses aga nii tihti ei tehta.

**Järjend** - andmetüüp, millega saame esitada loetelusid. Kõige lihtsam viis järjendit luua on kirjutada järjendisse kuuluvad väärtused komaga eraldatult nurksulgude vahele.

```
1 põhikooli_klassid = [1, 2, 3, 4, 5, 6, 7, 8, 9]
2
3 nimed = ["Tiit", "Teet", "Tuut"]
```

Järjendi `põhikooli_klassid` pikkus on 9 ja järjendi `nimed` pikkus on 3.

Järjenditega saab teha erinevaid operatsioone.

| **Tegevus** | **Avaldis** | **Väärtus** |
|---|---|---|
| Elementide arv järjendis | `len([1, 2, 3, 4, 5])` | `5` |
| Minimaalne element järjendis | `min([2, 1, 3])` | `1` |
| Maksimaalne element järjendis | `max([2, 1, 3])` | `3` |
| Elementide summa järjendis | `sum([2, 1, 3])` | `6` |
| Elementide järjestamine kasvavalt | `sorted([2, 1, 3])` | `[1, 2, 3]` |
| Elemendi sisaldumine järjendis | `3 in [2, 1, 3]` | `True` |
| Järjendite liitmine | `[2, 1] + [3, 1]` | `[2, 1, 3, 1]` |
| Elemendi esinemiste arv järjendis | `[2, 1, 1].count(1)` | `2` |

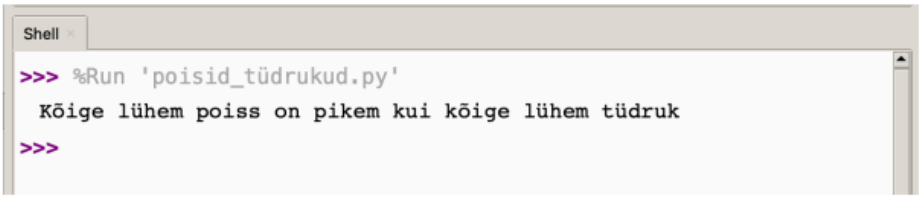
## ÜLESANNE 2

On antud kaks järjendit:

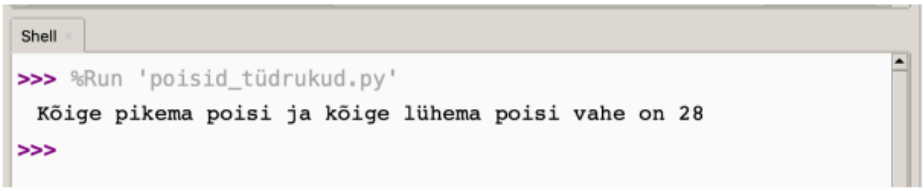
```
poiste_pikkused = [183, 188, 191, 174, 179, 178, 169, 195, 188, 185, 193, 178, 180, 179, 178, 179, 193, 197, 171]
tüdrukute_pikkused = [163, 158, 179, 175, 172, 179, 168, 165, 162, 169, 155, 172, 169, 167, 177, 172, 180, 175, 175, 176]
```

a. Kontrolli tingimuslause abil, kas poisse ja tüdrukuid on võrdselt. Kui ei ole, siis väljasta ekraanile, kumba on rohkem ("Poisse on rohkem kui tüdrukuid", "Tüdrukuid on rohkem kui poisse").
Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:


b. Leia kõige lühem poiss ja kõige lühem tüdruk. Võrdle tingimuslause abil, kas kõige lühem tüdruk on lühem kui kõige lühem poiss või vastupidi. Väljasta ekraanile, kumb on pikem ("Kõige lühem poiss on pikem kui kõige lühem tüdruk", "Kõige lühem tüdruk on pikem kui kõige lühem poiss").
Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:




c. Leia kõige pikema poisi ja kõige lühema poisi vahe.
Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:




d. Leia keskmise tüdruku pikkus.
Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:


e. Pane poiste ja tüdrukute pikkused uude järjendisse kokku. Leia, mitu poissi ja tüdrukut on pikkusega 172 ja 179 kokku.

Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:


f. Kontrolli, kas eksisteerib mõni tüdruk või poiss pikkusega 173.

Kui oled kõik õigesti teinud, siis käsurida peaks selline välja nägema:

## IX Praktikumi 6 töölehed

Piltide abil on esitatud praktikumi 6 töölehtede sisu.

# PRAKTIKUM 6

## JÄRJEND

### ÜLESANNE 1

On antud järjend `a = [8, 10, 4, 5, 6, 3, 1, 7, 2, 9]`.

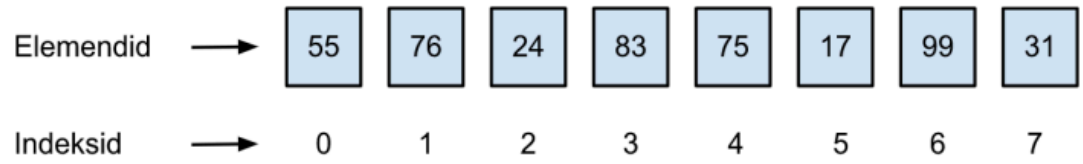
Kirjuta tabelisse, mis tulemused annavad järgmised read. Tee seda ülesannet peast!

| | |
|---|---|
| `min(a)` | |
| `max(a)` | |
| `sorted(a)` | |
| `sorted(a) + [11, 12]` | |
| `len(a)` | |
| `sum(a)` | |
| `5 in a` | |
| `15 in a` | |

Nüüd võta teist värvi kirjutusvahend ja eelmise nädala praktikumi lehtede alusel paranda oma vead.

**Pane tähele!** Enne selle nädala ülesannetega edasi liikumist lõpeta eelmise nädala praktikumi lehtedelt viimane ülesanne, kui see pooleli jäi.

Iga järjendi element asub mingil kindlal kohal - seda kohta näitab elemendi indeks.




**Pane tähele!** Programmeerimises ei loenda me alates ühest (1, 2, 3, 4, 5, …), vaid loendame alates nullist (0, 1, 2, 3, 4, …).

- See tähendab, et järjendis esimene element asub kohal 0 mitte 1.

Olgu meil mingi järjend `loomad = ["kass", "koer", "jänes", "lõvi"]`.

- Kui me soovime teada, mis indeksil asub väärtus "jänes", siis kirjutame `loomad.index("jänes")`. See tagastab meile vastuse 2 (väärtus "kass" asub indeksil 0 ja "koer" indeksil 1).
- Kui me soovime teada indeksil 3 asuvat väärtust, kirjutame lihtsalt `loomad[3]`. See tagastab meile vastuse "lõvi".

## ÜLESANNE 2

On antud järjend `a = [55, 76, 24, 83, 75, 17, 99, 31]`.

Kirjuta tabelisse, missugused koodiread annaks järgmisi vastuseid (kõik peavad olema seotud selle järjendiga).

Tee seda ülesannet peast!

| | |
|---|---|
| `99` | |
| `55` | |
| `17` | |
| `460` | |
| `[17, 24, 31, 55, 75, 76, 83, 99]` | |
| `True` | |
| `False` | |
| `8` | |

Nüüd võta teist värvi kirjutusvahend ja eelmise nädala praktikumi lehtede alusel paranda oma vead.

## X Praktikumi 7 töölehed

Piltide abil on esitatud praktikumi 7 töölehtede sisu.

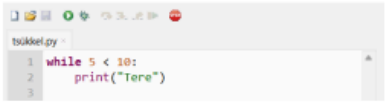
# PRAKTIKUM 7

## TSÜKKEL

Tihti on vaja programmi kirjutades mingit tegevust korrata. Selleks on meil tsükkel.

Üks levinumaid tsükleid on while-tsükkel.

while-tsüklis on ära defineeritud jätkamistingimus ja tsükli sisu. Kui jätkamistingimus on täidetud, tullakse tagasi jätkamistingimuse kontrolli juurde.




Antud tsükli puhul väljastatakse sõne "Tere" nii kaua, kuni arv 5 on väiksem kui arv 10. Et 5 on alati väiksem kui 10, siis tehakse seda igavesti.

| Jätkamistingimus | Sisu |
|---|---|
| Jätkamistingimus on tõeväärtustüüpi. Kui tingimus on täidetud, siis täidetakse tsükli sisu. Kui tingimus ei ole täidetud, siis tsükkel lõpeb ja tsükli sisu ei täideta. | Sisu on tegevus, mida soovime korrata, kuni jätkamistingimus kehtib. Sisu võib sisaldada erinevaid käske, muutujaid, tingimuslauseid ja ka teisi tsükleid. |


Esimese asjana anname muutujale i algväärtuse, milleks antud juhul on 0. Seejärel paneme paika jätkamistingimuse, milleks antud juhul on tingimus, et muutuja i on väiksem kui arv 10. See tähendab, et nii kaua, kuni muutuja i on vahemikus 0 kuni 9, siis tsükkel teeb oma tööd.

Tsükli sisus väljastatakse esimese asjana ekraanile sõna "Tere" ja seejärel suurendatakse muutuja i väärtust ühe võrra.

Kui tsükli sisu viimane osa on tehtud, siis minnakse tagasi tsükli tingimuse juurde ja kontrollitakse, kas muutuja i uue väärtuse juures tingimus veel kehtib - kas täita tsükli sisu veel või lõpetada.

**Pane tähele!** `i += 1` on sama mis `i = i + 1`.

**Pane tähele!** Samamoodi nagu tingimuslause korral, peab tsükli sisu olema taandega.

Järgmisel lehel on samm-sammult välja toodud eelmainitud tsükli läbimine koos tingimuste kontrolliga.

| | Muutuja i väärtus | Kas i < 10? | Tsükli sisu |
|---|---|---|---|
| 1. | i = 0 | Jah | Väljastatakse “Tere”.<br>i väärtust suurendatakse ühe võrra. |
| 2. | i = 1 | Jah | Väljastatakse “Tere”.<br>i väärtust suurendatakse ühe võrra. |
| 3. | i = 2 | Jah | Väljastatakse “Tere”.<br>i väärtust suurendatakse ühe võrra. |
| 4. | i = 3 | Jah | Väljastatakse “Tere”.<br>i väärtust suurendatakse ühe võrra. |
| 5. | i = 4 | Jah | Väljastatakse “Tere”.<br>i väärtust suurendatakse ühe võrra. |
| 6. | i = 5 | Jah | Väljastatakse “Tere”.<br>i väärtust suurendatakse ühe võrra. |
| 7. | i = 6 | Jah | Väljastatakse “Tere”.<br>i väärtust suurendatakse ühe võrra. |
| 8. | i = 7 | Jah | Väljastatakse “Tere”.<br>i väärtust suurendatakse ühe võrra. |
| 9. | i = 8 | Jah | Väljastatakse “Tere”.<br>i väärtust suurendatakse ühe võrra. |
| 10. | i = 9 | Jah | Väljastatakse “Tere”.<br>i väärtust suurendatakse ühe võrra. |
| 11. | i = 10 | Ei | Tsükli sisu ei täideta.<br>Tsükkel lõpetab töö. |

**Pane tähele!** Kui avastad, et programm käib juba tükk aega ja lõppu pole näha, siis Thonnys saab programmi töö lõpetada, vajutades STOP.

## ÜLESANNE 1

Vali õige vastus.

Kui while-tsükli jätkamistinigmuseks on avaldis 5 > 10, siis

a. tsükli sisu ei täideta kunagi.

b. tsükli sisu täidetakse lõpmatu arv kordi,

c. on tegemist vigase programmiga.

## VÄÄRTUSE MUUTMISE LÜHIVARIANDID

| | | |
|---|---|---|
| `a += 1` | `a = a + 1` | a väärtust suurendatakse ühe võrra |
| `a -= 1` | `a = a - 1` | a väärtust vähendatakse ühe võrra |
| `a *= 2` | `a = a * 2` | a väärtust suurendatakse kaks korda |
| `a /= 2` | `a = a / 2` | a väärtust vähendatakse kaks korda |
| `a //= 2` | `a = a // 2` | a väärtust vähendatakse kaks korda ja ümardatakse madalaima täisarvuni |
| `a %= 2` | `a = a % 2` | a väärtusest leitakse jääk kahega jagamisel |
| `a ** = 2` | `a = a ** 2` | a võetakse astmesse kaks |

Lahenda järgmised ülesanded järgmiselt:

- Analüüsi koodijuppi ja tõmba enda meelest õigele vastusevariandile ring ümber.
- Seejärel kirjuta need koodijupid ümber arvutisse ja vaata Thonny abil, mis on õige vastus.
    - Kui õige vastus erineb sellest, mis sina kirja panid, siis
        - paranda enda viga,
        - käi Thonny abil koodijupid ridahaaval läbi, et mõista, miks just selline vastus tuli.

## ÜLESANNE 2

Vali õige vastus. Mis ilmub ekraanile?

```
i = 0
while i < 5:
i = i + 1
print(i)
```

a. 0

b. 5

c. 6

d. Arvud 1 kuni 5 üksteise alla.

e. Veateade.

## ÜLESANNE 3

Vali õige vastus. Mis ilmub ekraanile?

```
i = 0
while i < 5:
      i = i + 1
print(i)
```

a. 0
b. 5
c. 6
d. Arvud 1 kuni 5 üksteise alla.
e. Veateade.

## ÜLESANNE 4

Vali õige vastus. Mis ilmub ekraanile?

```
i = 0
while i == 5:
      i = i + 1
print(i)
```

a. 0
b. 5
c. 6
d. Arvud 1 kuni 5 üksteise alla.
e. Veateade.

## ÜLESANNE 5

Vali õige vastus. Mis ilmub ekraanile?

```
i = 0
while i != 5:
      i = i + 1
print(i)
```

a. 0
b. 5
c. 6
d. Arvud 1 kuni 5 üksteise alla.
e. Veateade.

## ÜLESANNE 6

Vali õige vastus. Mis ilmub ekraanile?

```
i = 0
summa = 0
while summa < 5:
      summa = summa.+ 1
      i = i + 1
print(i)
```

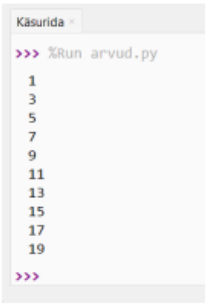
a. 0
b. 4
c. 5
d. 6
e. Arvud 1 kuni 5 üksteise alla.
f. Veateade.

**Pane tähele!** Tsükli sees saab väljastada ekraanile igal sammul muutuja i, kirjutades print(i).


## ÜLESANNE 7

Väljasta ekraanile kõik arvud vahemikus 1 kuni 20.

Kui oled kõik õigesti teinud, siis käsurida peaks nägema välja selline, nagu ülesandest vasakul näidatud on.




## ÜLESANNE 8

Väljasta ekraanile kõik paaritud arvud vahemikus 1 kuni 20.

Kui oled kõik õigesti teinud, siis käsurida peaks nägema välja selline, nagu ülesandest vasakul näidatud on.


## ÜLESANNE 9

Väljasta ekraanile kõik paarisarvud vahemikus 1 kuni 20.

Kui oled kõik õigesti teinud, siis käsurida peaks nägema välja selline, nagu ülesandest vasakul näidatud on.

### Turtle'is joonistamine tsükli abil

Kui tahaksime tsükli abil joonistada Turtle'is ruutu, peaksime selleks kirjutama järgmise programmi:

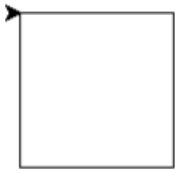
```
ruut.py
1 from turtle import *
2
3 i = 0
4
5 while i < 4:
6     forward(100)
7     right(90)
8     i += 1
```

Anname muutujale i algväärtuse 0 (tegelikult ei ole vahet, mis väärtuse me talle anname, peaasi, et see on tsükli jätkamistingimusega kooskõlastatud.

Seejärel paneme tsükli tingimuseks selle, et tegevused toimuksid täpselt 4 korda (muutuja i väärtused saavad olla 0, 1, 2, 3).

Iga kord, kui tsükli tingimus täidetakse, tahame liikuda edasi 100 pikslit ja keerata paremale 90 kraadi. Iga korraga peaksime liitma muutujale i ühe juurde, et peaksime järge, mitu korda oleme juba tsüklit täitnud.

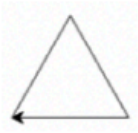
Selle tulemusena peaksime saama lõpptulemuseks ilusa ruudu.



## ÜLESANNE 10

Tuleta meelde, kuidas esimeses praktikumis Turtle'i abil joonistasid järgmised kujundid.

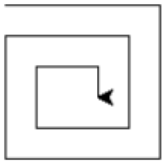
**Joonista need sel korral joonistada tsükli abiga**

1. 

2. 

## * ÜLESANNE 11

Joonista tsükli ja Turtle'i abil spiraal. Proovi teha võimalikult suur spiraal.



## XI Praktikumide 8 ja 9 tööleht

Pildi abil on esitatud praktikumide 8 ja 9 töölehe sisu.

# PRAKTIKUMID 8 JA 9

## TSÜKKEL

### ÜLESANNE 1

Kirjuta programm, mis küsib kasutajalt ühe naturaalarvu, ja väljastab ekraanile sõna "programmeerimine" see arv kordi.

### ÜLESANNE 2

Kirjuta programm, mis küsib kasutajalt parooli nii kaua, kuni kasutaja sisestab õige parooli. Kui kasutajal on õnnestunud õige parooli sisestamine, siis väljastatakse ekraanile lause "Sisenesid pangaautomaati!"

### ÜLESANNE 3

Palgid tahetakse laduda virna selliselt, et ülemises reas on üks palk ja igas järgmises reas on eelmisest ühe võrra rohkem palke (vaata pilti).
Kirjuta programm, kus kasutajalt küsitakse ridade arv ning arvutatakse ja väljastatakse ekraanile virnas olevate palkide arv.

Näiteks, kui arv on 5, siiis kuvatakse ekraanile 15, sest 1+2+3+4+5=15.
Arvude summa peab arvutama while-tsükli abil. Tsükli igal sammul tuleb liita eelmise sammuga võrreldes ühe võrra erinev arv.

### ÜLESANNE 4

Antud naturaalarvuga tehakse järgmini operatsioone:

- kui arv jagub 2-ga, siis see jagatakse 2-ga,
- kui arv ei jagu 2-ga, siis see korrutatakse 3-ga ja liidetakse 1.

Neid operatsioone tehes, peaks programm jõudma olenematata saadud naturaalarvust arvuni 1.
Kirjuta programm, mis küsib kasutajalt ühe naturaalarvu, ja väljastab ekraanile, mitu sammu kulub selleks, et jõuda vastavalt antud operatsioonidele arvuni 1.
Et olla kindel, kas programm käitub õigesti, võta enda sisestatud arv ja lahenda sellega need sammud käsitsi läbi ning vaata, mitu sammu saad.

## XII Praktikumide 10 ja 11 tööleht

Pildi abil on esitatud praktikumide 10 ja 11 töölehe sisu.

# PRAKTIKUMID 10-11

## TSÜKKEL

### ÜLESANNE 1

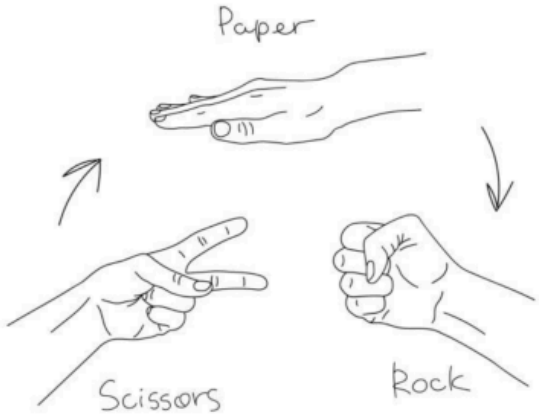
Kirjuta programm, mis annab kasutajale kümme võimalust arvata ära arv vahemikus 1 kuni 50. Iga arvamise korral annab progarmm kasutajale teada, kas

- kasutaja pakkus õige arvu,
- kasutaja pakkus õigest arvust suurema arvu,
- kasutaja pakkus õigest arvust väiksema arvu.

Peale kümmet katset peab programm kasutajale teada andma, et ta rohkem arvata ei saa ja mis oli õige arv.

### ÜLESANNE 2

Kirjutage programm, mis võistleb arvuti vastu mängus "kivi, paber, käärid".
Arvuti valik genereerida suvaliselt, kasutades random.randint, ja kasutajalt tuleks küsida tema valik.
Seejärel programm väljastab ekraanile arvuti valiku, kasutaja valiku ja tulemuse, kumb võitis.
Edasiarendus: Kirjuta programmile juurde osa, mis kontrolliks, kas kasutaja sisestas ühe kolmest valikust.
Näiteks, kui kasutaja teeb kirjavea ja kirjutab "kiivi" sõna "kivi" asemel, siis programm annab talle teada, et sõna tuleks uuesti sisestada.



## XIII Projekti nõuded

Pildi abil on esitatud tingimused, mille alusel töötati praktikumides 12–16.

# PRAKTIKUMID 12-16

PROJEKT

**Projekti tingimused:**

- Projekt peab olema ise tehtud. Projekt ei tohi olla internetist kopeeritud.
- Projekti kood peab olema kommenteeritud selliselt, et inimene, kes projekti ei teinud, saab kiiresti aru töö sisust.
- Projektis peab olema vähemalt üks tingimuslause.
- Projektis peab olema vähemalt üks tsükkel.
- Projektis tuleb kasutajalt andmeid küsida.
- Projektis tuleb ekraanile väljastada.
- Projekt peab sisaldama vähemalt ühte järjendit.

**Projektide ideid:**

- Bingo
- Trips-traps-trull
- Sõnaarvamusmäng (poomine)
- Laevade pommitamine

## XIV Testi teoreetilise osa ülesanded

Piltide abil on esitatud praktikumi 17 testi teoreetilise osa ülesanded.

# TEST

## TEOORIA

### PANE TÄHELE!

Töö tuleb teha iseseisvalt ilma materjalideta.Töö ajal suhtlemine on rangelt keelatud.

Töö sooritus ei mõjuta kursuse läbisaamist, vaid on informatsiooniks õpetajale.

Aega on 30 minutit.

### ÜLESANNE 1

On antud programm. Vali kõik õiged vastused.

```
1  sisend = int(input("Sisesta täisarv: "))
2  if sisend < 10:
3      if sisend > 5:
4          print("A")
5      else:
6          print("B")
7  else:
8      if sisend != 13:
9          print("C")
10     else:
11         print("D")
```

Millised read väljastatakse, kui kasutaja sisestab 3?

1. A
2. B
3. C
4. D
5. Mitte ükski

### ÜLESANNE 2

On antud programm. Vali kõik õiged vastused.

```
1  sisend = int(input("Sisesta täisarv: "))
2  if sisend < 10:
3      if sisend > 5:
4          print("A")
5      else:
6          print("B")
7  else:
8      if sisend != 13:
9          print("C")
10     else:
11         print("D")
```

Millised read väljastatakse, kui kasutaja sisestab 9?

1. A
2. B
3. C
4. D
5. Mitte ükski

### ÜLESANNE 3

On antud programm. Vali kõik õiged vastused.

```
1  sisend = int(input("Sisesta täisarv: "))
2  if sisend < 10:
3      if sisend > 5:
4          print("A")
5      else:
6          print("B")
7  else:
8      if sisend != 13:
9          print("C")
10     else:
11         print("D")
```

Millised read väljastatakse, kui kasutaja sisestab 13?

1. A
2. B
3. C
4. D
5. Mitte ükski

## ÜLESANNE 4

On antud programm. Vali kõik õiged vastused.

```
sisend = int(input("Sisesta täisarv: "))
if sisend < 10:
    if sisend > 5:
        print("A")
    else:
        print("B")
else:
    if sisend != 13:
        print("C")
    else:
        print("D")
```

Millised read väljastatakse, kui kasutaja sisestab 10?

1. A
2. B
3. C
4. D
5. Mitte ükski

## ÜLESANNE 5

On antud programm. Vali kõik õiged vastused.

```
sisend = int(input("Sisesta täisarv: "))
if sisend < 10:
    if sisend > 5:
        print("A")
    else:
        print("B")
else:
    if sisend != 13:
        print("C")
    else:
        print("D")
```

Millised read väljastatakse, kui kasutaja sisestab 5?

1. A
2. B
3. C
4. D
5. Mitte ükski

## ÜLESANNE 6

On antud programm.

```
järjend = [69, 9, 10, 12, 48, 60, 17, 12, 63, 93]

print(järjend.index(9))
```

Mis ilmub ekraanile?

Vastus: .....

## ÜLESANNE 7

On antud programm.

```
järjend = [69, 9, 10, 12, 48, 60, 17, 12, 63, 93]

print(järjend[6])
```

Mis ilmub ekraanile?

Vastus: .....

**ÜLESANNE 8**

On antud programm.

```
järjend = [69, 9, 10, 12, 48, 60, 17, 12, 63, 93]

print(järjend[3] + järjend[9])
```

Mis ilmub ekraanile?

Vastus: .....

**ÜLESANNE 9**

On antud programm.

```
järjend = [69, 9, 10, 12, 48, 60, 17, 12, 63, 93]

i = 0

while i < len(järjend):
    if järjend[i] % 2 == 0:
        print("Programmeerimine")
    i += 1
```

Mitu korda ilmub ekraanile sõna „Programmeerimine“?

Vastus: .....

**ÜLESANNE 10**

On antud programm.

```
järjend = [69, 9, 10, 12, 48, 60, 17, 12, 63, 93]

i = 0

while i < len(järjend):
    if järjend[i] % 2 == 0:
        print("Programmeerimine")
        i *= 2
    i += 1
```

Mitu korda ilmub ekraanile sõna „Programmeerimine“?

Vastus: .....

## XV Testi programmi koostamise ülesanne

Pildi abil on esitatud praktikumi 17 testi programmi koostamise ülesanne.

# TEST

## PROGRAMMI KOOSTAMINE

Õpetaja tahtis panna õpilastele hindeid skaalal 1 kuni 5. Kahjuks läks tal midagi valesti ja mõne hinde asemel läks hoopis 'x'. Õpetaja ei tahtnud, et keegi peaks tööd uuesti tegema ja võttis vastu otsuse, et arvutab korrektse hinde saanud õpilaste keskmise ja paneb sellest tulemusest 15% kõrgema hinde ümardatud täisarvuni nendele õpilastele, kellele läks hinde asemel 'x'.

Koostada programm, mis

- läbib järjendi, kus on etteantud hinded (läbitava järjendi nimi on „hinded"),
- leiab järjendist kõik arvulised väärtused,
- arvutab selles järjendis kõikide arvuliste väärtuste keskmise,
- leiab keskmisest 15% suurema arvu ja ümardab selle täisarvuni,
- tagastab ekraanile kirje: „Need, kelle hindeks läks 'x', saavad hindeks hoopis (siia tuleb sinu arvutatud hinne).".

Näide järjendist, kus asuvad õpilaste hinded:

```
hinded = [4, 5, 1, 2, 4, 'x', 4, 2, 1, 'x', 3, 4, 4, 2, 4, 5,
3, 5, 1, 5, 3, 1, 'x', 'x', 2, 5, 3, 5, 1, 1]
```

Sellisel juhul

- hinde saanud õpilaste keskmine hinne oleks ligikaudu 3,08,
- sellest 15% suurem arv on ligikaudu 3,54
- õpilased, kellel on hinde asemel 'x', saaksid hindeks 4.

## XVI Litsents

Mina, Hannaliina Rakovitš,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Programmeerimiskeele Python õpetamise töölehed III kooliastmele** jaoks, mille juhendajad on Tauno Palts ja Riin Saadjärv, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi Dspace kuni autoriõiguse kehtivuse lõppemiseni.
2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

**Hannaliina Rakovitš**
**10.08.2023**